西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

编程实现简单数据的输出（2）

1、printf函数

2、输出格式控制

3.2.2 格式化输出

- （3）flags标志字符
- 标志字符是一个字符，可以调整对齐、符号、空格以及八进制和十六进制前缀，格式说明中可以有多个标志字符。表3-3列出了标志字符的含义。

3.2.2 格式化输出

表3-3 printf标志字符含义

标志		意义	默认
-		在给定域宽内左对齐输出结果（右边用空格填充）	右对齐（左边用空格或0填充）
+		如果输出值是有符号数，则总是加上符号（+或-）	只在负数前加（-）
空格		如果输出值是有符号数或为正数，则以空格作为前缀加到输出值前；如果空格和+标志同时出现，则忽略空格	无

3.2.2 格式化输出

举例：

```
1  int a=123; double y=12;
2
3  //②输出带格式的整型数据
4  printf("[%+d],[%+d],[% d],[% d]\n",a,-a,a,-a);
5  //填充正负符号、填充空格
6  //输出结果：[+123],[-123],[ 123],[-123]
7  //⑦输出带格式的浮点型数据
8  printf("[%+lf],[%+lf],[% lf],[% lf]\n",y,-y,y,-y);
9  //填充正负符号、填充空格
10 //输出结果：[+12.000000],[-12.000000],[ 12.000000],[-12.000000]
```

3.2.2 格式化输出

续表3-3 printf标志字符含义

标志		意义	默认
#		指明使用如下的“转换样式”转换输出参数	
	若类型字符为	对输出参数的影响	
	x或X	在任何非0输出值前加上0x或0X	
	e，E，f	强制在所有情况下输出值总是包含小数点	只有小数点后面有数字时才显示它
	g，G	同e和E，强制在所有情况下输出值中总是包含小数点并阻止截断尾部的0	只有小数点后面有数字时才显示它，截断尾部的0

3.2.2 格式化输出

举例：

```
1 int a=123;
2
3 printf("%#d,%#x,%#X,%#o\n",a,a,a,a);
4 //填充十六进制、八进制前缀
5 //输出结果：123,0x7b,0X7B,0173
```

3.2.2 格式化输出

- （4）width宽度说明
- 宽度说明是非负的十进制整数，它规定输出占位的最小宽度。但输出大于宽度时，按实际值的输出，小的宽度不会引起输出值的截断。表3-4列出了宽度的含义。

3.2.2 格式化输出

表3-4 printf宽度说明

宽度说明	对输出域宽度的影响
n	至少有n个字符宽度输出，如果输出值中的宽度小于n个，则输出用空格填充直到最小宽度规定（如果flags 为一，则填充在输出值的右边，否则在左边）
0n	至少有n个字符宽度输出，如果输出值中的字符宽度小于n个，则输出用0填充在输出值的左边（对于左对齐无效）
*****	间接设置宽度，此时由输出项列表提供宽度值，且它必须在输出项的前面

3.2.2 格式化输出

举例：

```
1 int a=123,b=-1,c=12345;
2
3 //②输出带格式的整型数据
4 printf("[%d],[%4d],[%-4d],[%4d],[%-4d]\n",a,a,a,c,c);
5 //宽度、右对齐、左对齐、实际宽度
6 //输出结果：[123],[ 123],[123 ],[12345],[12345]
7 printf("[%04d],[%04d],[%04d],[%-04d]\n",a,b,c,a);
8 //左边填充0、右边不影响
9 //输出结果：[0123],[-001],[12345],[123 ]
```

3.2.2 格式化输出

举例：

```
1  char c1=97;
2
3  //④输出带格式的字符型数据
4  printf("[%12c],[%012c],[%-012c]\n",c1,c1,c1);
5  //宽度、右对齐、左对齐
6  //输出结果：[           a],[00000000000a],[a           ]
7  //⑧输出字符串
8  printf("[%s],[%6s],[%-6s]\n","Java","Java","Java");
9  //宽度对字符串的影响
10 //输出结果：[Java],[  Java],[Java  ]
```

3.2.2 格式化输出

举例：

```
1 int a=123;
2
3 printf("[%*d]\n",5,a); //由输出项指定宽度
4 //输出结果：[  123]
```

3.2.2 格式化输出

- （5）.prec精度说明
- 精度说明是以圆点（. ）开头的非负十进制整数，它规定了输出的最大字符数或有效数字位数。精度说明可以引起输出值的截断，或使浮点数输出值四舍五入。表3-5列出了精度的含义。

3.2.2 格式化输出

表3-5 printf精度说明

精度说明	精度影响	
.n	类型	含义
	e，E，f	精度值指定小数点后数字的个数。四舍五入
	g，G	精度值指定可输出的有效数字的最大数目
	s	精度值指定可输出字符的最大数目，超出精度值范围的字符不予输出
（无）	精度按默认值：	
	类型	默认值
	e，E，f	6
	g，G	打印6个有效数字，尾部的0串被截断
	s	输出直到空字符（'\0'）为止

3.2.2 格式化输出

续表3-5 printf精度说明

精度说明	精度影响	
.0或仅有.	类型	含义
	e，E，f，g，G	输出不打印小数点（及其后的小数）
	s	无任何字符输出
*		间接设置精度，此时由输出项列表提供精度值，且它必须在输出项的前面。如果宽度说明和精度说明同时使用*，则先出现宽度值，接着是精度值，然后才是输出项。

一个浮点类型值若是正无穷大、负无穷大或非IEEE浮点数时，printf函数输出＋INF、－INF、＋NAN或－NAN。

3.2.2 格式化输出

举例：

```
1  double x=12.3456,y=12;
2
3  //⑥输出指定精度的浮点型数据
4  printf("[%lf],[%10lf],[%10.2lf],[%.2lf]\n",x,x,x,x);
5  //默认精度、宽度、精度
6  //输出结果：[12.345600],[ 12.345600],[     12.35],[12.35]
7  printf("[%06.1lf],[%-06.1lf]\n",y,y);
8  //左边填充0、右边不影响
9  //输出结果：[0012.0],[12.0  ]
10 printf("[%.*f],[%*.*f]\n",6,x,12,3,x);
11 //由输出项指定宽度、宽度与精度
12 //输出结果：[12.345600],[      12.346]
```

3.2.2 格式化输出

举例：

```
1 int a=123;
2
3 printf("[%8.2d],[%-8.2d]\n",a,a); //精度对整型无作用
4 //输出结果：[      123],[123     ]
5 printf("[%s],[%.3s],[%6.3s]\n","Basic","Basic","Basic");
6 //精度对字符串的影响
7 //输出结果：[Basic],[Bas],[   Bas]
```

3.2.2 格式化输出

- （6）大小修饰
- 大小修饰指明输出结果的类型大小。表3-6列出了常用类型大小修饰的含义。

3.2.2 格式化输出

表3-6 printf类型大小修饰含义

大小修饰	type类型字符	输出参数被解释为
h	**d，o，x，X**	短整型（short）
	u	无符号短整型（unsigned short）
l	**d，o，x，X**	长整型（long）
	u	无符号长整型（unsigned long）
	e，E，f，g，G	双精度浮点型（double）
L	**e，E，f，g，G**	长双精度浮点型（long double）

3.2.2 格式化输出

举例：

```
1 long h=-1; short i=-1,j=32767;
2
3 //①输出整型数据
4 printf("%ld,%lu,%lx,%lo\n",h,h,h,h,h); //长整型，负数为补码
5 //输出结果：-1,4294967295,ffffffff,37777777777
6 printf("%hd,%hu,%hx,%ho\n",i,i,i,i,i); //短整型，负数为补码
7 //输出结果：-1,65535,ffff,177777
8 printf("%hd,%hd\n",j,j+1); //短整型，数据溢出
9 //输出结果：32767,-32768
```

3.2.2 格式化输出

例3.51

```
 1 #include <stdio.h>
 2 int main()
 3 {
 4    int a=123,b=-1,c=12345; long h=-1; short i=-1,j=32767;
 5    char c1=97; double x=12.3456,y=12,z=12.123456789123;
 6    //①输出整型数据
 7    printf("%d,%u,%x,%X,%o\n",a,a,a,a,a); //十进制、无符号、十六进制和八进制
 8    //输出结果：123,123,7b,7B,173
 9    printf("%d,%u,%x,%X,%o\n",b,b,b,b,b); //十进制、无符号、十六进制和八进制，负数为补码
10    //输出结果：-1,4294967295,ffffffff,FFFFFFFF,37777777777
11    printf("%ld,%lu,%lx,%lo\n",h,h,h,h,h); //长整型，负数为补码
12    //输出结果：-1,4294967295,ffffffff,37777777777
13    printf("%hd,%hu,%hx,%ho\n",i,i,i,i,i); //短整型，负数为补码
```

3.2.2 格式化输出

例3.51

```
14    //输出结果：-1,65535,ffff,177777
15    printf("%hd,%hd\n",j,j+1); //短整型，数据溢出
16    //输出结果：32767,-32768
17    //②输出带格式的整型数据
18    printf("[%d],[%4d],[%-4d],[%4d],[%-4d]\n",a,a,a,c,c);//宽
度、右对齐、左对齐、实际宽度
19    //输出结果：[123],[ 123],[123 ],[12345],[12345]
20    printf("[%+d],[%+d],[% d],[% d]\n",a,-a,a,-a); //填充正负
符号、填充空格
21    //输出结果：[+123],[-123],[ 123],[-123]
22    printf("[%04d],[%04d],[%04d],[%-04d]\n",a,b,c,a); //左边
填充0、右边不影响
23    //输出结果：[0123],[-001],[12345],[123 ]
24    printf("%#d,%#x,%#X,%#o\n",a,a,a,a); //填充十六进制、八进制
前缀
```

3.2.2 格式化输出

例3.51

```
25    //输出结果：123,0x7b,0X7B,0173
26    printf("[%*d]\n",5,a); //由输出项指定宽度
27    //输出结果：[  123]
28    printf("[%8.2d],[%-8.2d]\n",a,a); //精度对整型无作用
29    //输出结果：[     123],[123     ]
30    //③输出字符型数据
31    printf("%d,%c\n",c1,c1); //字符型数值、ASCII码
32    //输出结果：97,a
33    //④输出带格式的字符型数据
34    printf("[%12c],[%012c],[%-012c]\n",c1,c1,c1); //宽度、右对齐、左对齐
35    //输出结果：[           a],[00000000000a],[a           ]
36    //⑤输出浮点型数据
37    printf("%lf,%e,%g\n",x,x,x); //小数格式、指数格式、最简格式
38    //输出结果：12.345600,1.234560e+001,12.3456
```

3.2.2 格式化输出

例3.51

```
 39    printf("%lf,%e,%g\n",y,y,y); //小数格式、指数格式、最简格式
 40    //输出结果：12.000000,1.200000e+001,12
 41    //⑥输出指定精度的浮点型数据
 42    printf("[%lf],[%10lf],[%10.2lf],[%.2lf]\n",x,x,x,x);//默
认精度、宽度、精度
 43    //输出结果：[12.345600],[ 12.345600],[     12.35],[12.35]
 44    //⑦输出带格式的浮点型数据
 45    printf("[%+lf],[%+lf],[% lf],[% lf]\n",y,-y,y,-y); //填充
正负符号、填充空格
 46    //输出结果：[+12.000000],[-12.000000],[ 12.000000],[-12.00
0000]
 47    printf("[%06.1lf],[%-06.1lf]\n",y,y);//左边填充0,右边不影响
 48    //输出结果：[0012.0],[12.0  ]
 49    printf("[%.*f],[%*.*f]\n",6,x,12,3,x); //由输出项指定宽度、
宽度与精度
```

3.2.2 格式化输出

例3.51

```
50    //输出结果：[12.345600],[      12.346]
51    //⑧输出字符串
52    printf("[%s],[%6s],[%-6s]\n","Java","Java","Java"); //宽
度对字符串的影响
53    //输出结果：[Java],[  Java],[Java  ]
54    printf("[%s],[%.3s],[%6.3s]\n","Basic","Basic","Basic");
//精度对字符串的影响
55    //输出结果：[Basic],[Bas],[   Bas]
56    //⑨特殊输出
57    printf("%%\n",c1); //两个%%表示输出一个%，输出项
58    //输出结果：%
59    printf("%d,%d\n",a,b,c);//格式数目小于输出项数,忽略多余输出项
60    //输出结果：123,-1
61    printf("%d,%d,%d\n",a,b);//格式数目大于输出项数,输出结果不确定
62    //输出结果：123,-1,2367460
```

3.2.2 格式化输出

例3.51

```
63    printf("%d,%lf\n",x,a); //类型不对应，输出结果不确定
64    //输出结果：2075328197,0.000000
65    return 0;
66 }
```


程序设计