第七讲 输入输出与文件操作

潘建瑜@MATH.ECNU

目录页

Contents

C++ 基本输入输出

- 数据流
- 操纵符

I/O 基本概念

C++本身没有输入输出语句。

C++中的输入输出是通过相应的I/O流类库实现的。

数据流

将数据从一个对象到另一个对象的流动抽象为“流”

- 提取（读）：从流对象中获取数据，运算符为 ">>"
- 插入（写）：向流对象中添加数据，运算符为 "<<"

- 提取和插入运算符是所有标准 C++数据类型预先设计的
- 插入运算符与操纵符一起使用，可以控制输出格式

输入输出操作

- 头文件 iostream 中预定义的四个输入输出对象
  - cin：标准输入（键盘等）
  - cout：标准输出（屏幕、打印机等）
  - cerr：标准错误输出，没有缓冲，立即被输出
  - clog：与cerr类似，但有缓冲，缓冲区满时被输出
- 操纵符
  - 用于控制数据的输出格式
  - 两种途径：iomanip 头文件和 ios_base 类

操纵符

（需包含头文件 iomanip ）

操纵符	含义
endl	插入换行符，并刷新流
setw(n) cout.width(n)	设置域宽
setfill(字符) cout.fill(字符)	设置填充符
left / right	对齐方式（左对齐/右对齐），缺省右对齐
fixed	使用定点方式输出
scientific	使用指数形式
setprecision(n)	设置输出的有效数字个数； 若在 fixed 或 scientific 后使用，则设置小数位数
showpoint	显示小数点及尾随零，即使没有小数部分

操纵符（域宽）

- 设置宽度：setw(宽度) 或 cout.width(宽度)

```
int A[3][3]={{11,12,13},{21,22,23},{31,32,33}};
for (int i=0; i<3; i++)
   for (int j=0; j<3; j++)
     cout << setw(4) << A[i][j]; // 设置输出宽度
```

ex07_setw.cpp

```
for (int i=0; i<3; i++)
   for (int j=0; j<3; j++)
   {   cout.width(4);  // 设置输出宽度
       cout << A[i][j];
   }
```

  - setw 和 cout.width 只改变紧随其后的域，即只起一次作用
  - 若数据超过设置的宽度，则自动扩展到所需最少宽度

操纵符（填充符）

- 设置填充符：setfill(字符) 或 cout.fill(字符)

```
int A[3][3]={{11,12,13},{21,22,23},{31,32,33}};
// cout.fill('*'); // 设置填充符
cout << setfill('*'); // 设置填充符
for(int i=0; i<3; i++)
    for(int j=0; j<3; j++)
        cout << setw(4) << A[i][j]; // 设置输出宽度
```

ex07_fill.cpp

  - 缺省的填充符为空格
  - 该命令的作用将一直保留，直到下次改变为止

操纵符（对齐方式）

- 设置对齐方式：left / right

```
int A[3][3]={{11,12,13},{21,22,23},{31,32,33}};
cout << left; // 设置左对齐
for(int i=0; i<3; i++)
{
    for(int j=0; j<3; j++)
        cout << setw(4) << A[i][j];
    cout << "\n";
}
```

ex07_left.cpp

  - 缺省为右对齐
  - left/right 的作用是持久的，直到遇到下一个同类型命令

操纵符（浮点数输出格式）

- 以定点格式输出浮点数：fixed

```
cout << fixed; // 定点格式
for (int i=0; i<3; i++)
    cout << a[i];
```

- 以科学计数法（指数形式）输出浮点数：scientific

ex07_fixed_scientific.cpp

```
double a[3]={2.7182818, 31.416, 987000};
cout << scientific; // 以指数形式输出
for (int i=0; i<3; i++)
    cout << a[i];
```

操纵符（浮点数精度）

- 设置输出精度：setprecision

```
double a[3]={2.7182818, 31.416, 987000};
cout << setprecision(3);
for (int i=0; i<3; i++)
{
    cout << a[i];
}
```

ex07_setprecision.cpp

- 缺省为输出小数点后 6 位

建议	用单独一条语句设置全局类指令

操纵符等价用法

left	setiosflags(ios::left)
right	setiosflags(ios::right)
fixed	setiosflags(ios::fixed)
scientific	setiosflags(ios::scientific)

（更多操作符参见头文件 iomanip）

C 语言格式化输出

- printf 语句
- 格式字符串

C 语言格式化输出

- C 语言的输出函数：printf

需加头文件 cstdio

```
printf(“格式控制字符串”，输出变量列表)
```

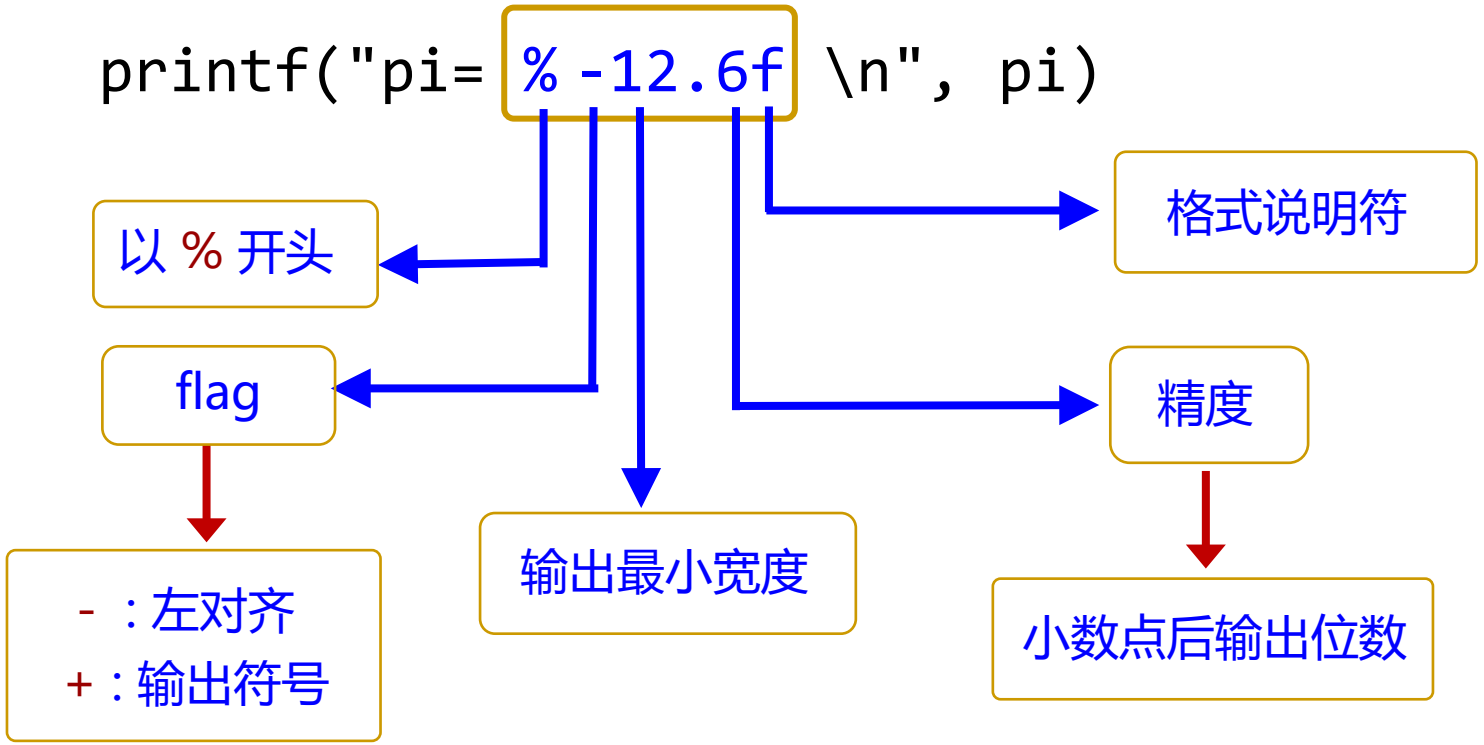
- 格式控制字符串：普通字符串、格式字符串、转义字符
  - 普通字符串：原样输出
  - 格式字符串：以 % 开头，后面跟各种 格式说明符
  - 转义字符：实现特殊功能

```
int k=5;
double a=3.14;
printf(“k=%d, a=%f\n”, k, a);
```

注：一个格式字符串对应一个输出变量！

格式字符串

%[flag][输出最小宽度][.精度]格式说明符



ex07_printf.cpp

格式说明符与转义字符

- 常见的格式说明符

c	字符型	g	浮点数（自动）
d	十进制整数	o	八进制
e	浮点数（科学计数法）	s	字符串
f	浮点数（小数形式）	x/X	十六进制

- 常见转义字符（输出特殊符号）

\b	退后一格	\t	水平制表符
\f	换页	\\	反斜杠
\n	换行	\"	双引号
\r	回车	%%	百分号

C 语言文件读写

- 文件指针
- 文件读写：
  打开文件、读写文件、关闭文件

文件分类

按存储介质

- 普通文件：存储介质文件（磁盘、磁带等）
- 设备文件：非存储介质（键盘、显示器、打印机等）

按数据的组织形式

- 文本文件：ASCII文件，每个字节存放一个字符的ASCII码
- 二进制文件：数据按其在内存中的存储形式原样存放

打开文件

- 文件指针

```
FILE *pf;
```

- 文件打开

```
pf=fopen(文件名，打开方式);
```

  - 文件名：普通字符串
  - 打开方式：读、写、文本文件、二进制文件

rt、wt、at、rb、wb、ab、rt+、wt+、at+、rb+、wb+、ab+

(r 为读，w 为写，+ 为读写，t 为文本，b 为二进制)

注：若文件打开成功，返回指向文件的指针，否则返回一个空指针 (NULL)

关闭文件

- 文件关闭

```
fclose(pf);
```

  - 正常关闭返回值为 0; 出错时，返回值为非0

```
FILE *pf;
pf=fopen("out1.txt","wt");
if (pf==NULL)
{
   printf("ERROR: Can not open the file!\n");
   exit(1);
}
fprintf(pf,"This is my first file.\n");
fclose(pf);
```

文件读写：文本文件

- 写文本文件

```
fprintf(pf, "格式控制字符串", 输出变量列表);
```

  - fprintf 用法与 printf 类似

ex07_fprintf.cpp

```
FILE *pf;
pf=fopen("out1.txt","wt");
double pi=3.1415926;
fprintf(pf,"pi=%-12.6f\n",pi);
fclose(pf);
```

- 读文本文件

```
fscanf(pf, "格式控制字符串", 地址列表);
```

```
fscanf(fp,"%d,%f", &i, &t);
```

文件读写举例：文本文件

ex07_fscanf.cpp

```
double A[]={1.0/3, 1.0/6, 1.0/7, 1.0/9, 1.0/11};
FILE * pf;
pf = fopen("out.txt","wt");
for(int i=0; i<n; i++)
   fprintf(pf, "%.6f\n", A[i]);  // 建议只写数据
fclose(pf);

double x[n];
pf = fopen("out.txt","rt");
for(int i=0; i<n; i++)
   fscanf(pf, "%lf", x+i);   // 要使用 %lf
fclose(pf);
```

文件读写：二进制文件

- 写二进制文件

```
fwrite(buffer, size, count, pf);
```

  - 将 count 个长度为 size 的连续数据写入到 pf 指向的文件中，buffer 是这些数据的首地址（可以是指针或数组名）

- 读二进制文件

```
fread(buffer, size, count, pf);
```

  - 从 pf 指向的文件中读取 count 个长度为 size 的连续数据，buffer 是存放这些数据的首地址（可以是指针或数组名）

文件读写：二进制文件

ex07_fwrite_fread.cpp

```
int A[3][3]={{11,12,13},{21,22,23},{31,32,33}};
FILE *pf;
pf=fopen("data1.dat","wb");
fwrite(A,sizeof(int),9,pf);  // 也可以写为 fwrite(A,sizeof(A),1,pf);
fclose(pf);

int B[3][3];
pf=fopen("data1.dat","rb");
fread(B,sizeof(int),9,pf);
fclose(pf);
```

第七讲上机作业

1、生成一个 6X6 的矩阵 ***A***，其元素为 **[0, 1]** 之间的随机双精度数。

(a) 将其按矩阵形式写入到一个文本文件 **out71.txt** 中；

(b) 将其写入到一个二进制文件 **data71.dat** 中；

(c) 从二进制文件 **data71.dat** 中读取前 **12** 个数据（双精度），放到一个 2X6 的矩阵 ***B*** 中，并将 ***B*** 按行输出。（程序取名 **hw07_01.cpp** ）

2、从课程主页上下载二进制数据文件 **data72.dat**，从文件中读取前 **60** 个元素（双精度），构成一个 30X2 的矩阵 ***A***。然后将其按矩阵形式写入到一个文本文件 **out72.txt** 中。（程序取名 **hw07_02.cpp** ）

第七讲上机作业

3、计算一个正整数的所有数字之和。

编写两个函数, 分别用 循环 和 递归 计算一个整数的所有数字之和,

并在主函数中分别调用这两个函数计算 **2012112118** 的所有数字之和。（程序取名 **hw07_03.cpp** ）

4、多项式乘积运算：编写函数, 计算两个多项式 $a(x)$ 和 $b(x)$ 的乘积 $c(x) = a(x) \times b(x)$

(a) 多项式用其系数所组成的数组来表示, 比如 $a(x) = 2x^3 - 6x + 1$ 对应的数组为 [2, 0, -6, 1];

(b) 函数原型见下面, 其中 `pa`, `pb`, `pc` 分别是指向数组 a, b, c 的指针,

这里 a, b, c 是数组, 分别代表多项式 $a(x)$, $b(x)$, $c(x)$;

`m` 和 `n` 分别是数组 a 和 b 的长度;

在主函数中调用该函数计算多项式 $a(x) = 2x^3 - 6x + 1$ 和 $b(x) = 3x^2 + x - 2$ 的乘积。（程序取名 **hw07_04.cpp** ）

```
void poly_prod(double * pa, double * pb, double * pc, int m, int n);
```

5、（可选题）编写函数, 实现求解线性方程组的 Gauss 消去法（见讲义练习 6.10）。

（程序取名 **hw07_05.cpp** ）