C++

Programming

输入输出流

Input Output Streams

2025年4月21日

学而不厌 诲人不倦

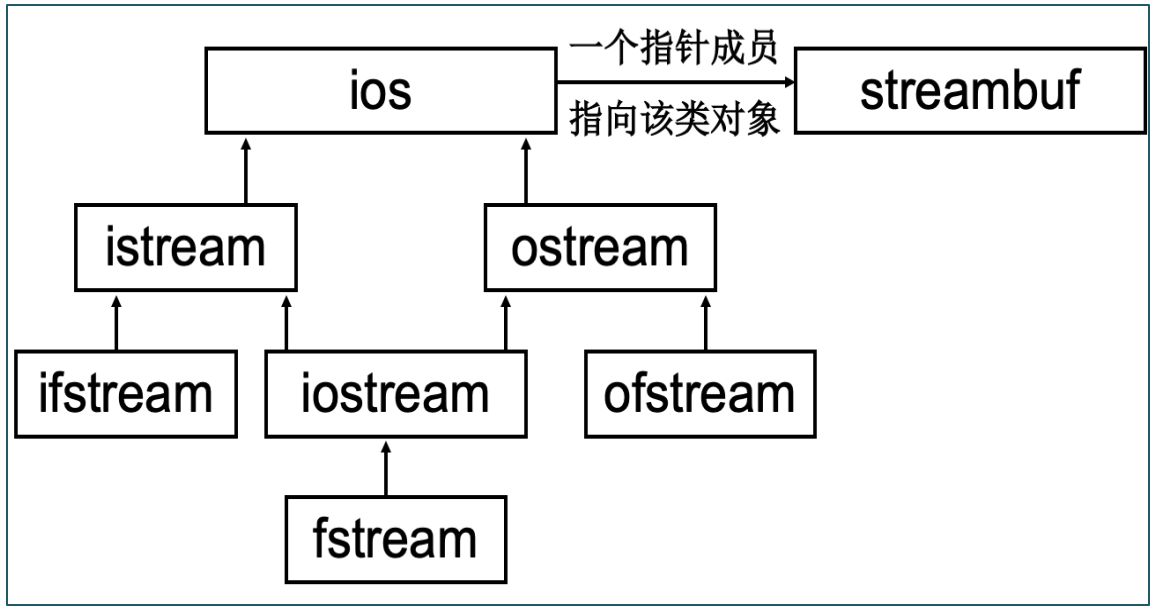
Chapter 7 输入输出流

- 7.1 输入和输出
- 7.2 标准输出流
- 7.3 标准输入流
- 7.4 文件操作与文件流

7.1 输入和输出

1. 流类库

- 在C++中，输入输出是通过**流**完成的。把接收存放输出数据的地方叫做**目标**，把输入数据来自的地方叫做**源**，输入和输出操作可以看成字符序列在源、目标和对象之间的流动。
- C++把与输入和输出有关的操作定义为一个类体系。这个执行输入和输出操作的类体系叫做**流类**，提供流类实现的系统叫做**流类库**。



流类库的基本模板类等级图

- **ios**类是istream类和ostream类的虚基类，ios类中一个指针成员指向streambuf类的对象。streambuf类管理流的缓冲区。
- **istream**：常用于接收从键盘输入的数据；
- **ostream**：常用于将数据输出到屏幕上；
- **iostream**：继承自 istream 和 ostream 类，因为该类的功能兼两者于一身，既能用于输入，也能用于输出；
- **ifstream**：用于读取文件中的数据；
- **ofstream**：用于向文件中写入数据；
- **fstream**：兼 ifstream 和 ofstream 类功能于一身，既能读取文件中的数据，又能向文件中写入数据。

2. C++输入流和输出流

- 在C++中，cin 就是 istream 类的对象，cout 是 ostream 类的对象
- <iostream> 头文件中还声明有 2 个 ostream 类对象，分别为 cerr 和 clog。cerr 常用来输出警告和错误信息给程序的使用者，clog 常用来输出程序执行过程中的日志信息

```
#include <iostream>
#include <string>
int main() {
    std::string url;
    std::cin >> url;
    std::cout << "cout: " << url << std::endl;
    std::cerr << "cerr: " << url << std::endl;
    std::clog << "clog: " << url << std::endl;
    return 0;
}
```

7.1 输入和输出

➢ 2. C++输入流和输出流

表 1 C++ cin 输入流对象常用成员方法

成员方法名	功能
getline(str,n,ch)	从输入流中接收 n-1 个字符给 str 变量，当遇到指定 ch 字符时会停止读取，默认情况下 ch 为 '\0'。
get()	从输入流中读取一个字符，同时该字符会从输入流中消失。
gcount()	返回上次从输入流提取出的字符个数，该函数常和 get()、getline()、ignore()、peek()、read()、readsome()、putback() 和 unget() 联用
peek()	返回输入流中的第一个字符，但并不是提取该字符。
putback(c)	将字符 c 置入输入流（缓冲区）。
ignore(n,ch)	从输入流中逐个提取字符，但提取出的字符被忽略，不被使用，直至提取出 n 个字符，或者当前读取的字符为 ch。
operator>>	重载 >> 运算符，用于读取指定类型的数据，并返回输入流对象本身。

表 2 C++ cout 输出流对象常用成员方法

成员方法名	功能
put()	输出单个字符。
write()	输出指定的字符串。
tellp()	用于获取当前输出流指针的位置。
seekp()	设置输出流指针的位置。
flush()	刷新输出流缓冲区。
operator<<	重载 << 运算符，使其用于输出其后指定类型的数据。

```
int main() {
char url[30] = {0};
cin.getline(url, 30); //读取一行字符串
//输出上一条语句读取字符串的个数
cout << "读取了 "<<cin.gcount()<<" 个字符" << endl;
cout.write(url, 30); //输出 url 数组存储的字符串
return 0;
}
```

7.2 标准输出流

➢ 1. cout成员方法格式化输出

表 1 ostream 类的成员方法

成员函数	说明
flags(fmtfl)	当前格式状态全部替换为 fmtfl。注意，fmtfl 可以表示一种格式，也可以表示多种格式。
precision(n)	设置输出浮点数的精度为 n。
width(w)	指定输出宽度为 w 个字符。
fill(c)	在指定输出宽度的情况下，输出的宽度不足时用字符 c 填充（默认情况是用空格填充）。
setf(fmtfl, mask)	在当前格式的基础上，追加 fmtfl 格式，并删除 mask 格式。其中，mask 参数可以省略。
unsetf(mask)	在当前格式的基础上，删除 mask 格式。

7.2 标准输出流

➢ 1. cout成员方法格式化输出

```
#include <iostream>
using namespace std;
int main()
{
    double a = 1.23;
    //设定后续输出的浮点数的精度为 4
    cout.precision(4);
    cout <<"precision: "<< a << endl;
    //设定后续以科学计数法的方式输出浮点数
    cout.setf(ios::scientific);
    cout <<"scientific: "<< a << endl;
    return 0;
}
```

表 2 fmtfl 和 mask 参数可选值

标 志	作 用
ios::boolapha	把 true 和 false 输出为字符串
ios::left	输出数据在本域宽范围内向左对齐
ios::right	输出数据在本域宽范围内向右对齐
ios::internal	数值的符号位在域宽内左对齐，数值右对齐，中间由填充字符填充
ios::dec	设置整数的基数为 10
ios::oct	设置整数的基数为 8
ios::hex	设置整数的基数为 16
ios::showbase	强制输出整数的基数（八进制数以 0 开头，十六进制数以 0x 打头）
ios::showpoint	强制输出浮点数的小点和尾数 0
ios::uppercase	在以科学记数法格式 E 和以十六进制输出字母时以大写表示
ios::showpos	对正数显示“+”号
ios::scientific	浮点数以科学记数法格式输出
ios::fixed	浮点数以定点格式（小数形式）输出
ios::unitbuf	每次输出之后刷新所有的流

7.2 标准输出流

➢ 1. cout成员方法格式化输出

```
#include <iostream>
using namespace std;
int main()
{
    double f = 123;
    //设定后续以科学计数法表示浮点数
    cout.setf(ios::scientific);
    cout << f << '\n';
    //删除之前有关浮点表示的设定
    cout.unsetf(ios::floatfield);
    cout << f;
    return 0;
}
```

表 2 fmtfl 和 mask 参数可选值

标 志	作 用
ios::boolapha	把 true 和 false 输出为字符串
ios::left	输出数据在本域宽范围内向左对齐
ios::right	输出数据在本域宽范围内向右对齐
ios::internal	数值的符号位在域宽内左对齐，数值右对齐，中间由填充字符填充
ios::dec	设置整数的基数为 10
ios::oct	设置整数的基数为 8
ios::hex	设置整数的基数为 16
ios::showbase	强制输出整数的基数（八进制数以 0 开头，十六进制数以 0x 打头）
ios::showpoint	强制输出浮点数的小点和尾数 0
ios::uppercase	在以科学记数法格式 E 和以十六进制输出字母时以大写表示
ios::showpos	对正数显示“+”号
ios::scientific	浮点数以科学记数法格式输出
ios::fixed	浮点数以定点格式（小数形式）输出
ios::unitbuf	每次输出之后刷新所有的流

7.2 标准输出流

➢ 2. 使用流操纵算子格式化输出

表 3 C++ 流操纵算子

流操纵算子	作 用	
*dec	以十进制形式输出整数	常用
hex	以十六进制形式输出整数	
oct	以八进制形式输出整数	
fixed	以普通小数形式输出浮点数	
scientific	以科学计数法形式输出浮点数	
left	左对齐，即在宽度不足时将填充字符添加到右边	
*right	右对齐，即在宽度不足时将填充字符添加到左边	
setbase(b)	设置输出整数时的进制，b=8、10 或 16	
setw(w)	指定输出宽度为 w 个字符，或输入字符串时读入 w 个字符。注意，该函数所起的作用是一次性的，即只影响下一次 cout 输出。	
setfill(c)	在指定输出宽度的情况下，输出的宽度不足时用字符 c 填充（默认情况是用空格填充）	
setprecision(n)	设置输出浮点数的精度为 n。 在使用非 fixed 且非 scientific 方式输出的情况下，n 即为有效数字最多的位数，如果有效数字位数超过 n，则小数部分四舍五入，或自动变为科学计 数法输出并保留一共 n 位有效数字。 在使用 fixed 方式和 scientific 方式输出的情况下，n 是小数点后面应保留的位数。	
setiosflags(mask)	在当前格式状态下，追加 mask 格式，mask 参数可选择表 2 中的所有值。	
resetiosflags(mask)	在当前格式状态下，删除 mask 格式，mask 参数可选择表 2 中的所有值。	
boolapha	把 true 和 false 输出为字符串	不常用
*noboolalpha	把 true 和 false 输出为 0、1	
showbase	输出表示数值的进制的前缀	
*noshowbase	不输出表示数值的进制.的前缀	
showpoint	总是输出小数点	
*noshowpoint	只有当小数部分存在时才显示小数点	
showpos	在非负数值中显示 +	

➢ 2. 使用流操纵算子格式化输出

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    //以十六进制输出整数
    cout << hex << 16 << endl;
    //删除之前设定的进制格式，以默认的 10 进制输出整数
    cout << resetiosflags(ios::basefield)<< 16 << endl;
    double a = 123;
    //以科学计数法的方式输出浮点数
    cout << scientific << a << endl;
    //删除之前设定的科学计数法的方法
    cout << resetiosflags(ios::scientific) << a << endl;
    return 0;
}
```

7.2 标准输出流

3. 输出单个字符和字符串

- **put() 方法专用于向输出流缓冲区中添加单个字符**
- **ostream&put(char c);**
- cout.put(c1).put(c2).put(c3);
- **write() 成员方法专用于向输出流缓冲区中添加指定的字符串**
- **ostream&write（const char * s，streamsize n）；**
- cout.write(c1, 1).write(c2, 2).write(c3, 3);

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    string str = "ten.gnehcnaib.c//:ptth";
    for (int i = str.length() - 1; i >= 0; i--) {
    cout.put(str[i]); //从最后一个字符开始输出
    }
    cout.put('\n');
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main() {
    cout.write("http://", 7)
    .write("c.biancheng.net", 15)
    .write("/cplus/", 7);
    return 0;
}
```

7.3 标准输入流

➢ 1. cin输入流对象常用成员方法

表 1 C++ cin 输入流对象常用成员方法

成员方法名	功能
getline(str,n,ch)	从输入流中接收 n-1 个字符给 str 变量，当遇到指定 ch 字符时会停止读取，默认情况下 ch 为 '\0'。
get()	从输入流中读取一个字符，同时该字符会从输入流中消失。
gcount()	返回上次从输入流提取出的字符个数，该函数常和 get()、getline()、ignore()、peek()、read()、readsome()、putback() 和 unget() 联用。
peek()	返回输入流中的第一个字符，但并不是提取该字符。
putback(c)	将字符 c 置入输入流（缓冲区）。
ignore(n,ch)	从输入流中逐个提取字符，但提取出的字符被忽略，不被使用，直至提取出 n 个字符，或者当前读取的字符为 ch。
operator>>	重载 >> 运算符，用于读取指定类型的数据，并返回输入流对象本身。

2. 读取单个字符

- get()函数从输入流中读入一个字符，返回值就是该字符的 ASCII 码。如果碰到输入的末尾，则返回值为 EOF。
- EOF 是 End of File 的缩写。istream 类中从输入流（包括文件）中读取数据的成员函数，在把输入数据都读取完后再进行读取，就会返回 EOF。
- EOF 是在 iostream 类中定义的一个整型常量，值为 -1。
- get() 函数不会跳过空格、制表符、回车等特殊字符，所有的字符都能被读入。

```
#include <iostream>
using namespace std;
int main()
{
    int c;
    while ((c = cin.get()) != EOF)
    cout.put(c);
    return 0;
}
```

2. 读入一行字符串

- **istream & getline(char* buf, int bufSize);**
- 从输入流中读取 bufSize-1 个字符到缓冲区 buf，或遇到\n为止（哪个条件先满足就按哪个执行）。函数会自动在 buf 中读入数据的结尾添加\0。
- **istream & getline(char* buf, int bufSize, char delim);**
- 读到 delim 字符为止。\n或 delim 都不会被读入 buf，但会被从输入流中取走。
- 两个函数的返回值就是函数所作用的对象的引用。如果输入流中\n或 delim 之前的字符个数达到或超过 bufSize，就会导致读入出错，其结果是：虽然本次读入已经完成，但是之后的读入都会失败。
- cin >> str在碰到行中的空格或制表符时就会停止，因此就不能保证 str 中读入的是整行。

7.3 标准输入流

➢ 2. 读入一行字符串

```
#include <iostream>
using namespace std;
int main()
{
    char szBuf[20];
    int n = 120;
    if(!cin.getline(szBuf,6)) //如果输入流中一行字符超过5个，就会出错
    cout << "error" << endl;
    cout << szBuf << endl;
    cin >> n;
    cout << n << endl;
    cin.clear(); //clear能够清除cin内部的错误标记，使之恢复正常
    cin >> n;
    cout << n << endl;
    return 0;
}
```

- 7.1 输入和输出
- 7.2 标准输出流
- 7.3 标准输入流
- 7.4 文件操作与文件流

1. 读入单个字符

- 功能：将文本文件 test.txt 中的全部内容原样显示出来

```
#include <iostream>
using namespace std;
int main()
{
    int c;
    freopen("test.txt", "r", stdin); //将标准输入重定向为 test.txt
    while ((c = cin.get()) != EOF)
    cout.put(c);
    return 0;
}
```

2. 读入一行字符串

- 可以用 getline() 函数的返回值（为 false 则输入结束）来判断输入是否结束。例如，要将文件 test.txt 中的全部内容（假设文件中一行最长有 10 000个字符）原样显示

```
#include <iostream>
using namespace std;
const int MAX_LINE_LEN = 10000; //假设文件中一行最长 10000 个字符
int main()
{
    char szBuf[MAX_LINE_LEN + 10];
    freopen("test.txt", "r", stdin); //将标准输入重定向为 test.txt
    while (cin.getline(szBuf, MAX_LINE_LEN + 5))
    cout << szBuf << endl;
    return 0;
}
```

3. cout.tellp()成员方法

- tellp() 成员方法用于获取当前输出流缓冲区中最后一个字符所在的位置
- **streampos tellp();**
- tellp() 不需要传递任何参数，会返回一个 streampos 类型值。事实上，streampos 是 fpos 类型的别名，而 fpos 通过自动类型转换，可以直接赋值给一个整形变量（即 short、int 和 long）。也就是说，在使用此函数时，我们可以用一个整形变量来接收该函数的返回值。

3. cout.tellp()成员方法

```
#include <iostream> //cin 和 cout
#include <fstream> //文件输入输出流
int main() {
    //定义一个文件输出流对象
    std::ofstream outfile;
    //打开 test.txt，等待接收数据
    outfile.open("test.txt");
    const char * str = "http://c.biancheng.net/cplus/";
    //将 str 字符串中的字符逐个输出到 test.txt 文件中，每个字符都会暂时存在输出流缓冲区中
    for (int i = 0; i < strlen(str); i++) {
    outfile.put(str[i]);
    //获取当前输出流
    long pos = outfile.tellp();
    std::cout << pos << std::endl;
    }
    //关闭文件之前，刷新 outfile 输出流缓冲区，使所有字符由缓冲区流入test.txt文件
    outfile.close();
    return 0;
}
```

4. cout.seekp()成员方法

- seekp() 方法用于指定下一个进入输出缓冲区的字符所在的位置。
- **ostream& seekp (streampos pos);** //指定下一个字符存储的位置
- //通过偏移量间接指定下一个字符的存储位置
- **ostream& seekp (streamoff off, ios_base::seekdir way);**
- seekp() 方法会返回一个引用形式的 ostream 类对象，意味着 seekp() 方法可以这样使用：
- `cout.seekp(23) << "当前位置为: " << cout.tellp();`

➢ 4. cout.seekp()成员方法

```
#include <iostream> //cin 和 cout
#include <fstream> //文件输入输出流
using namespace std;
int main() {
    ofstream outfile; //定义一个文件输出流对象
    outfile.open("test.txt"); //打开 test.txt，等待接收数据
    const char * str = "http://c.biancheng.net/cplus/";
    //将 str 字符串中的字符逐个输出到 test.txt 文件中，每个字符都会暂时存在输出流缓冲区中
    for (int i = 0; i < strlen(str); i++) {
    outfile.put(str[i]);
    }
    cout << "当前位置为：" << outfile.tellp() << endl; //获取当前输出流
    //调整新进入缓冲区字符的存储位置
    outfile.seekp(23); //等价于：//outfile.seekp(23, ios::beg);//outfile.seekp(-6, ios::cur);
    //outfile.seekp(-6, ios::end);
    cout << "新插入位置为:" << outfile.tellp() << endl;
    const char* newstr = "python/";
    outfile.write("python/", 7);
    outfile.close(); //关闭文件之前，刷新 outfile 输出流缓冲区，使所有字符由缓冲区流入test.txt文件
    return 0;
}
```

Chapter 7 输入输出流

- 7.1 输入和输出
- 7.2 标准输出流
- 7.3 标准输入流
- 7.4 文件操作与文件流

SpotLight3 自选题目

1. 围绕课程内容设计自选题目，题目和内容应能体现个人最高能力和水平；
2. 认真准备**演讲PPT和Spotlight视频**；
3. 梳理课程主要知识点，可绘制思维导图，**撰写课程总结并提出建议**，形式不限，字数不低于1000字；
4. 截止日期前将如下三个文件一起提交到知新平台：

   (1) Spotlight3 视频 (2) 演讲PPT (3) 课程总结和建议

Thank You !

Q & A