C++之 IO 流

输入流/输出流

IO 流的状态

文件流

字符串流

C++标准库对 IO 设备进行了抽象，统一使用流的方式对他们进行操作，同时定义了其他类型，例如字符串流，使得 string 对象能够像文件一样操作

前面涉及的 IO 工具有：

istream　　输入流

ostream　　输出流

cin、cout、cerr

>>操作符

<<操作符

getline 整行读取字符串

流的概念：

可以把流看做“水管”，这样输入流指的是从程序外部输入数据，类似于水通过水管流进程序，输出流恰好相反，是从程序内部输出到

外部。这里要注意的是，输入输出的参照物是程序。

典型的输出流是 cin，输出流是 cout 和 cerr。

流的分类：

C++中的 IO 流一共有三种，分别是：

控制台

cin cout cerr

磁盘文件

ifstream

ofstream

fstream

字符串流

istringstream

ostringstream

stringstream

流的特性：

IO 对象不可复制或赋值。

这里有两个含义：一是 IO 对象不能放在 vector 中，而是形参或者返回类型不能为流类型。

所以在使用 IO 流做参数传递或者返回值的时候，统一采用引用的形式，而且一半是非 const 的。

IO 流的条件状态：

考虑下面的程序：

```
int ival;
std::cin >>ival;
```

如果输入 3.14 会发生什么？后面还能继续输入吗？

如果 cin 接收到不符合类型的输入数据，则会读取失败，此时 cin 无法继续读入数据，进入了一个错误状态

如果想继续输入，必须重新设置 cin 流

这就涉及到了 IO 流的三种状态：

bad　系统级故障，不可恢复

fail 可以恢复的错误

eof　cin 碰到了文件结尾

同时系统提供了一系列函数来查询 IO 流的状态

查询状态

```
s.eof()
s.fail()
s.bad()
s.good()      //s 处于有效状态
```

设置状态

```
s.clear()
s.clear(flag)
```

s.setstate(flag)
s.rsstate()

看下面的程序：

```
int a;
if (cin.good()) {
    cout << "cin is good!" << endl;
}
while (cin >> a) {
    cout << a << endl;
}
if (cin.eof()) {
    cout << "eof!" << endl;
}
if (cin.fail()) {
    cout << "fail!" << endl;
}
std::string s;
cin >> s;
cout << s << endl;
```

对于这个程序，我们分别输入如下：

1. 3/4/^D 后面输入 test
2. hello 后面输入 test

我们会看到，对于第一种输入，打印的是：good/eof/fail 以及后面输入的 test，而对于第二个输入，仅仅输出了 good/fail，后面的 test 没有打印出来。

原因是因为：

Cin 接收到非法数据，会导致输入失败，同时 cin 本身不可用！

如果 cin 接收到类型不匹配的数据，则 fail 状态失败，如果接收到 ^D，那么 fail 和 eof 都会失败！

那么流失败的情况下想要继续输入应该怎么办？应该进行修复，示例代码如下：

```
int ival;
using std::cin;
while (cin >> ival, !cin.eof()) {
    if (cin.bad())
        throw std::runtime_error("IO stream corrupted");
    if (cin.fail()) {
        std::cerr << "bad data, try again!" << std::endl;
        cin.clear();    重置状态
        cin.ignore(std::numeric_limits < std::streamsize > ::max(), '\n');        忽略错误输入
        continue;
    }
}
```

当然需要导入相应的头文件

输出缓冲区：

每个 IO 对象都有一个缓冲区，手工刷新缓冲区的办法有：

```
std::cout << "hi" << std::flush;
std::cout << "hi" << std::ends;
std::cout << "hi" << std::endl;
```

在程序中应该多使用 endl，而不是’\n’

文件流

文件流有三种类型

ifstream

ofstream

fstream

打开文件的方式：

两种方式：

```
std::iftream is(“in.txt”)
std::ifstream is;
is.open(filename.c_str());
```

如何从文件中读取文本内容？

```
std::ifstream is;
is.open("in.txt");

std::string word;
std::vector<std::string> vec;
while(is >> word)
{
    vec.push_back(word);
}
```

文件最后要关闭，最好进行 clear 重置状态

从文本进行整行读入：

```
while (std::getline(is, line))
```

如何检测文件是否打开成功？

我们采用下面这个典型的程序：

```
std::ifstream &open_file(std::ifstream &in, const std::string &file) {
    in.close();
    in.clear();
    in.open(file.c_str());
    return in;
}
```

使用方式就是这样：

```
if (!open_file(is, filename)) {
        throw std::runtime_error("file open failed!");
```

```
    }
```

字符串流：

字符串流其实就是把输入输出的对象由文件改为 string，字符串流有下面三种：

istringstream

ostringstream

stringstream

字符串流的使用方式如下：

```
string line, word;
while(getline(cin, line))
{
    istringstream stream(line);
    while(stream >> word)
    {
        //process
    }
}
```

下面编写程序，将文件的每一行存储在 vector<string>中，然后将每个单词存储在另一个 vector<string>中。

代码如下：

```
    std::ifstream & read_file(const std::string &filename, std::ifstream &is,
        std::istringstream &iss, std::vector<std::string> &lines,
        std::vector<std::string> &words) {

    if (!open_file(is, filename)) {
        throw std::runtime_error("file open failed!");
    }
    std::string line;
    std::string word;
    while (std::getline(is, line)) {
        lines.push_back(line);
```

```
            iss.str(line);
            while (iss >> word) {
                words.push_back(word);
            }
            iss.clear();
        }
        is.close();
        is.clear();
        return is;
    }
```