# **c++文件的读取和写入**



[mengdicfm](https://me.csdn.net/mengdicfm" \t "/Users/qianrongsheng/Documents\\x/_blank) 2018-09-30 15:59:39 IMG_257 2919 IMG_258 收藏 11

分类专栏： [C++](https://blog.csdn.net/mengdicfm/category_8042701.html" \t "/Users/qianrongsheng/Documents\\x/_blank)

ofstream是从内存到硬盘，ifstream是从硬盘到内存，其实所谓的流缓冲就是内存空间

在C++中，有一个stream这个类，所有的I/O都以这个“流”类为基础的，包括我们要认识的文件I/O.

stream这个类有两个重要的运算

1、插入器(<<)  
　　向流输出数据。比如说系统有一个默认的标准输出流(cout)，一般情况下就是指的显示器，所以，cout<<“Write Stdout”<<’\n’;就表示把字符串"Write Stdout"和换行字符(’\n’)输出到标准输出流。  
　　2、析取器(>>)  
　　从流中输入数据。比如说系统有一个默认的标准输入流(cin)，一般情况下就是指的键盘，所以，cin>>x;就表示从标准输入流中读取一个指定类型的数据。  
　　在C++中，对文件的操作是通过stream的子类fstream(file stream)来实现的，所以，要用这种方式操作文件，就必须加入头文件fstream.h。

常见的文件操作:1

一、打开文件  
　　在fstream类中，有一个成员函数open()，就是用来打开文件的，其原型是：  
　　void open(const char\* filename,int mode,int access);

参数：  
　　filename：　　要打开的文件名  
　　mode：　　　　要打开文件的方式  
　　access：　　　打开文件的属性  
　　打开文件的方式在类ios(是所有流式I/O类的基类)中定义.

常用的值如下：  
　　ios::app：　　　以追加的方式打开文件  
　　ios::ate：　　　文件打开后定位到文件尾，ios:app就包含有此属性  
　　ios::binary：　以二进制方式打开文件，缺省的方式是文本方式。两种方式的区别见前文  
　　ios::in：　　　 文件以输入方式打开(文件数据输入到内存)  
　　ios::out：　　　文件以输出方式打开(内存数据输出到文件)  
　　ios::nocreate： 不建立文件，所以文件不存在时打开失败  
　　ios::noreplace：不覆盖文件，所以打开文件时如果文件存在失败  
　　ios::trunc：　　如果文件存在，把文件长度设为0

可以用“或”把以上属性连接起来，如ios::out|ios::binary  
　　打开文件的属性取值是：  
　　0：普通文件，打开访问  
　　1：只读文件  
　　2：隐含文件  
　　4：系统文件  
　　可以用“或”或者“+”把以上属性连接起来，如3或1|2就是以只读和隐含属性打开文件。  
　　例如：以二进制输入方式打开文件c:\config.sys  
　　fstream file1;  
　　file1.open(“c:\config.sys”,ios::binary|ios::in,0);  
　　如果open函数只有文件名一个参数，则是以读/写普通文件打开，即：  
　　file1.open(“c:\config.sys”); <=> file1.open(“c:\config.sys”,ios::in|ios::out,0);  
　　另外，fstream还有和open()一样的构造函数，对于上例，在定义的时侯就可以打开文件了：  
　　fstream file1(“c:\config.sys”);

特别提出的是，fstream有两个子类：

* 1

ifstream(input file stream)和ofstream(outpu file stream)，

ifstream默认以输入方式打开文件

ofstream默认以输出方式打开文件。  
　　ifstream file2(“c:\pdos.def”);//以输入方式打开文件  
　　ofstream file3(“c:\x.123”);//以输出方式打开文件

所以，在实际应用中，根据需要的不同，选择不同的类来定义：

如果想以输入方式打开，就用ifstream来定义;

如果想以输出方式打开，就用ofstream来定义;

如果想以输入/输出方式来打开，就用fstream来定义。  
　　二、关闭文件  
　　打开的文件使用完成后一定要关闭，fstream提供了成员函数close()来完成此操作，

如：file1.close();

就把file1相连的文件关闭。  
　　三、读写文件  
　　读写文件分为文本文件和二进制文件的读取.

对于文本文件的读取比较简单，用插入器和析取器就可以了;

对于二进制的读取就要复杂些，下要就详细的介绍这两种方式  
　　1、文本文件的读写  
　　文本文件的读写很简单：

用插入器(<<)向文件输出;

用析取器(>>)从文件输入。

假设file1是以输入方式打开，file2以输出打开。

示例如下：  
　　file2<<“I Love You”;//向文件写入字符串"I Love You"

int i;  
　　file1>>i;//从文件输入一个整数值。  
　　这种方式还有一种简单的格式化能力，比如可以指定输出为16进制等等，具体的格式有以下一些  
　　操纵符 功能 输入/输出  
　　dec 格式化为十进制数值数据 输入和输出  
　　endl 输出一个换行符并刷新此流 输出  
　　ends 输出一个空字符 输出  
　　hex 格式化为十六进制数值数据 输入和输出  
　　oct 格式化为八进制数值数据 输入和输出  
　　setpxecision(int p) 设置浮点数的精度位数 输出  
示例:  
#include   
#include   
using namespace std;

void main( void )  
{  
//利用ofstream类的构造函数创建一个文件输出流对象来打开文件  
ofstream fout( “d:\mytest.txt” );  
if ( ! fout)  
{  
cout << “文件不能打开” <<endl;  
}  
else  
{  
// 输出到磁盘文件  
fout << “Learning C++ is very useful.”<< endl;  
//关闭文件输出流  
fout.close();  
//利用ifstream类的构造函数创建一个文件输入流对象  
ifstream fin( “d:\mytest.txt” );  
if ( ! fin)  
{  
cout << “文件不能打开” <<endl;  
}  
else  
{  
char buffer[80];  
// 从磁盘文件输入  
fin >> buffer;  
// 关闭文件输入流  
fin.close();  
cout << buffer << endl;  
}  
}  
ofstream file1;  
if (!file1)  
{  
cout << “文件不能打开” <<endl;  
}  
else  
{  
//open函数只有文件名一个参数，则是以读/写普通文件打开  
file1.open(“d:\mytest.txt”);  
[//相当于file1.open](https://xn--file1-r41hu51hss3b.open/)(“d:\mytest.txt”,ios::out|ios::in,0);  
//文本文件的写:用插入器(<<)向文件输出.  
file1<<“写入文件操作!”<<endl;  
file1.close();  
}  
ifstream file2(“d:\mytest.txt”);  
if (!file2)  
{  
cout << “文件不能打开” <<endl;  
}  
else  
{  
char ch1[100];  
//文本文件的读:用析取器(>>)从文件输入.  
file2>>ch1;  
cout<<ch1<<endl;  
file2.close();  
}  
}  
　　2、二进制文件的读写  
　　①put()  
　　put()函数向流写入一个字符，其原型是ofstream &put(char ch)，使用也比较简单，如file1.put(‘c’);就是向流写一个字符’c’。  
　　②get()  
　　get()函数比较灵活，有3种常用的重载形式：  
　一种就是和put()对应的形式：ifstream &get(char &ch);功能是从流中读取一个字符，结果保存在引用ch中，如果到文件尾，返回空字符。如file2.get(x);表示从文件中读取一个字符，并把读取的字符保存在x中。  
　　另一种重载形式的原型是： int get();这种形式是从流中返回一个字符，如果到达文件尾，返回EOF，如x=file2.get();和上例功能是一样的。  
　　还有一种形式的原型是：ifstream &get(char \*buf,int num,char delim=’\n’);这种形式把字符读入由 buf 指向的数组，直到读入了 num 个字符或遇到了由 delim 指定的字符，如果没使用 delim 这个参数，将使用缺省值换行符’\n’。例如：  
　　file2.get(str1,127,‘A’); //从文件中读取字符到字符串str1，当遇到字符’A’或读取了127个字符时终止。  
　　③读写数据块  
　　要读写二进制数据块，使用成员函数read()和write()成员函数，它们原型如下：  
　　read(unsigned char \*buf,int num);  
　　write(const unsigned char \*buf,int num);  
　　read()从文件中读取 num 个字符到 buf 指向的缓存中，如果在还未读入 num 个字符时就到了文件尾，可以用成员函数 int gcount();来取得实际读取的字符数;而 write() 从buf 指向的缓存写 num 个字符到文件中，值得注意的是缓存的类型是 unsigned char *，有时可能需要类型转换。  
　　例：  
　　unsigned char str1[]=“I Love You”;  
　　int n[5];  
　　ifstream in(“[xxx.xxx](http://xxx.xxx/)”);  
　　ofstream out(“yyy.yyy”);  
　　out.write(str1,strlen(str1));[//把字符串str1全部写到yyy.yyy中](https://xn--str1yyy-k53ky2v9nas7cd79csyuby2flh6c.xn--yyy-x68d/)  
　　in.read((unsigned char*)n,sizeof(n));[//从xxx.xxx中读取指定个整数](https://xn--xxx-8p9d.xn--xxx-l68doa808fv7nrooz8eeb6251g/)，注意类型转换  
　　in.close();out.close(); 四、检测EOF

成员函数eof()用来检测是否到达文件尾，如果到达文件尾返回非0值，否则返回0。原型是int eof();  
　　例：　　if(in.eof()) ShowMessage(“已经到达文件尾!”);  
　　五、文件定位  
　　和C的文件操作方式不同的是，C++ I/O系统管理两个与一个文件相联系的指针。一个是读指针，它说明输入操作在文件中的位置;另一个是写指针，它下次写操作的位置。每次执行输入或输出时，相应的指针自动变化。所以，C++的文件定位分为读位置和写位置的定位，对应的成员函数是seekg()和seekp()。seekg()是设置读位置， seekp是设置写位置。它们最通用的形式如下：  
　　istream &seekg(streamoff offset,seek\_dir origin);  
　　ostream &seekp(streamoff offset,seek\_dir origin);  
　　streamoff定义于 iostream.h 中，定义有偏移量 offset 所能取得的最大值，seek\_dir 表示移动的基准位置，是一个有以下值的枚举：  
　　ios::beg：　　文件开头  
　　ios::cur：　　文件当前位置  
　　ios::end：　　文件结尾  
　　这两个函数一般用于二进制文件，因为文本文件会因为系统对字符的解释而可能与预想的值不同。例：  
　　file1.seekg(1234,ios::cur); //把文件的读指针从当前位置向后移1234个字节  
　　file2.seekp(1234,ios::beg); //把文件的写指针从文件开头向后移1234个字节

c++中输出和输入导屏幕和键盘的类别声明包含再标题文件<iostrream.h>中，而磁盘类文件的 I/O则声明再包含标题文件<fstream.h>内。

输入和输出格式：

输出到磁盘 ofsteam 识别字（“文件名”）

从磁盘读文件 ifsteam 识别字（"文件名“）

例如：

ofstream outfile(“data.txt”); [//写入到磁盘的data.txt中](https://xn--data-ps8fvko0bz94quvaf2m.xn--txt-x68d/)

格式化输入输出：

1 整数数据的输入输出

整数数据存储再磁盘内，每个文字各占一个字节。

例如：

#include <fstream.h>

#include <iostream.h>

#inlude <conio.h>

void main()

{

ofstream outfile(“data.txt”); //写入文件

for(int i=0;i<10;i++)

outfile<<i<<" "; //空格是为了避免数值连接在一起加上去的，此文件大小为20字节

cout<<“ok,press a key!”;

gerch();  
}

程序执行后用记事本打开可以看到数据

0 1 2 3 4 5 6 7 8 9

同样读取文件语句为：

int data;

ifstream infile(“data.txt”);

for(int i=0;i<10;i++)

{

infile>>data ; //读数据的时候因为数据间有一个空格才能完整的读出，

cout<<data<<" "; //输出到屏幕，加上空格是为了分开数字

}

2 字符数据的输入

字符的输出方式以put(ch)为存入语句，读取语句为get(ch)

例如：

char str[]=“hello this is a c++ programe!\n”

“i know this difference \n”

“and mocive”;

ofstream outfile (“data.txt”);

for(int i=0;i<strlen(Str);i++)

outfile.put(str[i]);

读取：

char ch;

ifstream infile(“data.txt”);

while(infile)

{

infile.get(ch);

cout<<ch;

}

getch();

}

3 字符串数据的输入

数据多的时候读写速度比较快，输入时以整行字符串加上换行符号一次写入。读取的时候以语句getline(buffer,max)，来读取整行数据，直到遇到换行符，每行结尾的\n并不读入，所以在 输出的时候需要加上换行符号，否则数据会连接在一起。

例子：

ofstream outfile(“strdata.txt”);

outfile<<“use your vote\n”；

outfile<<“ouse your weise\n”;

读取：

const MAX=80;

char buffer[MAX];

ifstream infile(“strdata.txt”);

while(infile)

{

infile.getline(buffer,MAX);

cout<<buffer<<endl;

}

4 浮点数

浮点数因为有小数点，在存储数据时与整数相同，只要每个数据加上一个空格就可以区隔相邻的数据

6 二进制文件

上述的格式化文件比较占用硬盘控件，采用二进制存储就可以节约很多控件。它使用write,read()来存储和读取。

ofstream 识别字（”文件名“，ios::binary);

write( 写入地址，写入大小）

ifstream 识别字（”文件名“，ios:binary);

识别字.read(读取地址，读取大小）；

例如：infile.read((char\*)buffer,sizeof(数据类型));

关闭文件

识别字.close();

例子：

ofstream outfile(“data.dat”,ios::binary);

for(int i=0;i<100;i++)

{

n[0]=i+1;

outfile.write((Char\*)n,sizeof(int)); //还可以用变量和数组为例

// outfile.write((Char\*)&i,sizeof(int)); i 为变量

// outfile.write((Char\*)n,sizeof(int)); n为数组

}

outfile.close();

ifstream inifile(“data.dat”,ios:binary);

for(i=0;i<100;i++)

{ inifile.read((Char\*)n,sizeof(int);

cout<<n[0]<<" ";

}

7 随机存取文件

文件能够随意读出，读出后又可以更新，更新后可以回存到源文件内。fstream file ;

file.open(“文件名”,存取模式）；

file.open(“x.dat”,ios::app|ios::in|ios::out|ios::binary);

in 打开文件输入或者读取 ifstream

out 打开文件输出或者写入 ofstream

ate 从文件尾开始写入或者读取

app 加在文件尾

arunc 若文件存在，讲其长度设为0

binary 打开二进制文件

二进制文件中有一个指针，指向当前数据在文件中的位置，这个文件指针和一般的指针变量不一样，它只是一个纯粹的指示器。

函数介绍

seekg()

seekg(0) 指针移到文件的最前面

seekg(0,ios::cur);把当前的指针当作0

seekg(0,ios::end);将指针移到文件尾，若再配合file.tellg()则可以求出文件的大小为多少bytes

以下两个操作都必须在文件关闭后才可以使用

remove("文件名”）；把这个文件删除

rename(“旧文件名”,“新文件名”);

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

 int n[5] = {1, 2, 3, 4, 5};

 register int i;

 ofstream out("test", ios::out | ios::binary);

 if(!out) {

 cout << "Cannot open file.\n";

 return 1;

 }

 out.write((char \*) &n, sizeof n);

 out.close();

 for(i = 0; i <5; i++) // clear array

 n[i] = 0;

 ifstream in("test", ios::in | ios::binary);

 if(!in) {

 cout << "Cannot open file.\n";

 return 1;

 }

 in.read((char \*) &n, sizeof n);

 for(i = 0; i <5; i++) // show values read from file

 cout << n[i] << " ";

 in.close();

 return 0;

}

---------------------

本文来自 黄常宇的马达加斯加 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/doraemon\_\_\_/article/details/53349754?utm\_source=copy

1. 逐行读入

void readTxt(string file)

{

ifstream infile;

infile.open(file.data()); //将文件流对象与文件连接起来

assert(infile.is\_open()); //若失败,则输出错误消息,并终止程序运行

string s;

while(getline(infile,s))

{

cout<<s<<endl;

}

infile.close(); //关闭文件输入流

}

1. 逐个字符读入

void readTxt(string file)

{

ifstream infile;

infile.open(file.data()); //将文件流对象与文件连接起来

assert(infile.is\_open()); //若失败,则输出错误消息,并终止程序运行

string s;

while(getline(infile,s))

{

cout<<s<<endl;

}

infile.close(); //关闭文件输入流

}