

2019年春

# 程序设计实习(I): C++程序设计

## 第十二讲 文件操作

刘家瑛

*liujiaying@pku.edu.cn*



# 文件操作

- 数据的层次
- 文件和流
- 建立顺序文件
- 文件的读写指针
- 有格式读写
- 无格式读写



# 数据的层次

- 位 bit
- 字节 byte
- 域 / 记录

例：学生记录

```
int ID;
```

```
char name[10];
```

```
int age;
```

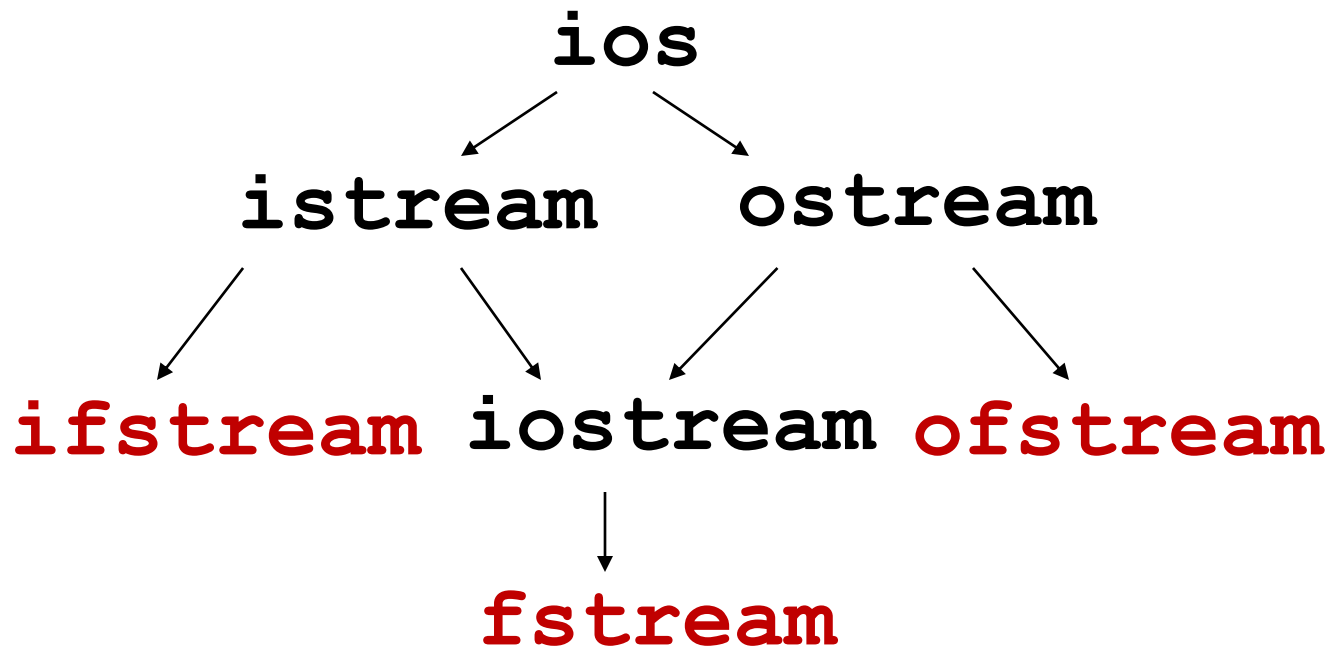
```
int rank[10];
```

- 将所有记录顺序地写入一个文件, 称为 顺序文件



# 文件和流

- 将顺序文件看作一个有限字符构成的顺序字符流  
然后像对**cin/cout**一样的读写
- 回顾一下输入输出流类的结构层次：



# 建立顺序文件

`#include <fstream>` // 包含头文件

`ofstream` outFile("clients.dat", ios::out|ios::binary); // 打开文件

- ofstream 是 fstream 中定义的类
- outFile 是我们定义的 ofstream 类的对象
- "clients.dat" 是将要建立的文件的文件名
- ios::out 是打开并建立文件的选项
  - ios::out 输出到文件, 删除原有内容
  - ios::app 输出到文件, 保留原有内容, 总是在尾部添加
  - ios::ate 输出到文件, 保留原有内容, 在文件任意位置添加
- ios::binary 以二进制文件格式打开文件



# 建立顺序文件

- 也可以先创建ofstream对象, 再用**open函数**打开  
**ofstream fout;**

```
fout.open("test.out", ios::out|ios::binary);
```

- 判断打开是否成功:

```
if (!fout) { cerr << "File open error!" <<endl; }
```

文件名可以给出绝对路径, 也可以给相对路径

没有交代路径信息, 就是在当前文件夹下找文件



# 文件的读写指针

- 对于输入文件, 有一个**读指针**
- 对于输出文件, 有一个**写指针**
- 对于输入输出文件, 有一个**读写指针**
- 标识文件操作的当前位置, 该指针在哪里, 读写操作就在哪里进行



# 文件的读写指针

```
ofstream fout("a1.out", ios::ate);
```

```
long location = fout.tellp();
```

//取得写指针的位置

```
location = 10L;
```

```
fout.seekp(location);
```

// 将写指针移动到第10个字节处

```
fout.seekp(location, ios::beg); //从头数location
```

```
fout.seekp(location, ios::cur); //从当前位置数location
```

```
fout.seekp(location, ios::end); //从尾部数location
```

//location 可以为负值





# 文件的读写指针

```
ifstream fin("a1.in", ios::ate);
```

```
long location = fin.tellg();
```

//取得读指针的位置

```
location = 10L;
```

```
fin.seekg(location);
```

// 将读指针移动到第10个字节处

```
fin.seekg(location, ios::beg); //从头数location
```

```
fin.seekg(location, ios::cur); //从当前位置数location
```

```
fin.seekg(location, ios::end); //从尾部数location
```

//location 可以为负值



# 字符文件读写

- 因为文件流也是流, 所以前面讲过的流的成员函数和流操作算子也同样适用于文件流
- 写一个程序, 将文件 `in.txt` 里面的整数排序后, 输出到 `out.txt`

例, 如 **`in.txt`** 的内容为:

1 234 9 45 6 879

2 4

则执行本程序后, 生成的 **`out.txt`** 的内容为:

1 2 4 6 9 45 234 879

假定待排序的数不超过1000个



# 参考程序

```
#include "iostream"
#include "fstream"
#include "algorithm"
using namespace std;
int aNum[1000];
int main() {
    ifstream srcFile("in.txt", ios::in);
    ofstream destFile("out.txt", ios::out);
    int x;
    int n = 0;
    while( srcFile >> x )
        aNum[n++] = x;
    sort(aNum, aNum + n); //定义在头文件<algorithm>
    for( int i = 0; i < n; i ++ )
        destFile << aNum[i] << " ";
    destFile.close();
    srcFile.close();
}
```



# 二进制文件读写

```
int x=10;  
fout.seekp(20, ios::beg);  
fout.write((char *)(&x), sizeof(int));  
  
fin.seekg(0, ios::beg);  
fin.read((char *)(&x), sizeof(int));
```

- 二进制文件读写, 直接写二进制数据, 记事本看未必正确



# 二进制文件读写

//下面的程序从键盘输入几个学生的姓名的成绩,并以二进制  
//文件形式存起来

```
#include "iostream"  
#include "fstream"  
using namespace std;
```

```
class CStudent {  
    public:  
        char szName[20];  
        int nScore;  
};
```



```
int main()
{
    CStudent s;
    ofstream OutFile( "c:\\tmp\\students.dat", ios::out|ios::binary);
    while( cin >> s.szName >> s.nScore ) {
        if( strcmp(s.szName, "exit") == 0) //名字为exit则结束
            break;
        OutFile.write( (char * ) & s, sizeof(s) );
    }
    OutFile.close();
    return 0;
}
```



输入:

Tom 60

Jack 80

Jane 40

exit 0

**Note -- 文本文件/二进制文件打开文件的区别:**

- 在Unix/Linux下, 二者一致, 没有区别;
- 在Windows下, 文本文件是以“\r\n”作为换行符  
→ 读出时, 系统会将0x0d0a只读入0x0a  
→ 写入时, 对于0x0a系统会自动写入0x0d

则形成的 students.dat 为 72字节, 用记事本打开, 呈现:

Tom 烫烫烫烫烫烫烫烫< Jack 烫烫烫烫烫烫烫烫藥 Jane 烫烫烫烫  
烫烫烫?



# 二进制文件读写

//下面的程序将 students.dat 文件的内容读出并显示

```
#include <iostream>
#include <fstream>
using namespace std;
class CStudent {
public:
    char szName[20];
    int nScore;
};
```





```
int main() {  
    CStudent s;  
    ifstream InFile("c:\\tmp\\students.dat", ios::in);  
    if(!InFile) {  
        cout << "error" << endl;  
        return 0;  
    }  
    while( InFile.read( (char* ) & s, sizeof(s) ) ) {  
        int nReadedBytes = InFile.gcount(); //看刚才读了多少字节  
        cout << s.szName << " " << s.nScore << endl;  
    }  
    InFile.close();  
    return 0;  
}
```

输出：  
Tom 60  
Jack 80  
Jane 40



# 二进制文件读写

//下面的程序将 students.dat 文件的Jane的名字改成Mike

```
#include <iostream>
#include <fstream>
using namespace std;
class CStudent {
public:
    char szName[20];
    int nScore;
};
```



```
int main(){
    CStudent s;
    fstream iofile( "c:\\tmp\\students.dat", ios::in|ios::out);
    if( !iofile) {
        cout << "error" ;
        return 0;
    }
    iofile.seekp( 2 * sizeof(s), ios::beg); //定位写指针到第三个记录
    iofile.write("Mike", strlen("Mike"));
    iofile.seekg(0, ios::beg); //定位读指针到开头
    while( iofile.read( (char* ) & s, sizeof(s)) )
        cout << s.szName << " " << s.nScore << endl;
    iofile.close();
    return 0;
}
```

**输出：**  
**Tom 60**  
**Jack 80**  
**Mike 40**



# 显式关闭文件

- `ifstream fin("test.dat", ios::in);`  
`fin.close();`
- `ofstream fout("test.dat", ios::out);`  
`fout.close();`



# 命令行参数

## □ 启动程序运行时可以指定命令行参数

### ■ 方式一

- 开始 → 运行 → 输入cmd, 按确定 → dos窗口
- 可以用 cd 切换当前目录: .. cd 子目录 等等
- 来到程序所在目录, 启动程序: myprogram arg1 arg2
- 或者在任意目录下, 启动程序: e://me//aa arg1 arg2
- 绝对路径和相对路径的概念

### ■ 方式二

- 在VC编程环境中: project -> settings->debug ->program arguments



# 命令行参数

```
int main(int argc, char* argv){  
    cout << argc << endl;  
    for(int i=0; i<argc; i++)  
        cout << argv[i] << endl;  
    .....  
}
```

第一个参数为命令本身, 第二个以后为参数



# 例：mycopy 程序，文件拷贝

/\*用法示例：

**mycopy src.dat dest.dat**

即将 **src.dat** 拷贝到 **dest.dat**

如果 **dest.dat** 原来就有, 则原来的文件会被覆盖

\*/

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main(int argc, char * argv[])
```

```
{
```

```
    if( argc != 3 ) {
```

```
        cout << "File name missing!" << endl;
```

```
        return 0;
```

```
}
```



```
ifstream inFile(argv[1], ios::binary|ios::in); //打开文件用于读
if ( ! inFile ) {
    cout << "Source file open error." << endl;
    return 0;
}
ofstream outFile(argv[2], ios::binary|ios::out); //打开文件用于写
if ( !outFile ) {
    cout << "New file open error." << endl;
    inFile.close(); //打开的文件一定要关闭
    return 0;
}
char c;
while ( inFile.get(c) ) //每次读取一个字符
    outFile.put(c); //每次写入一个字符
outFile.close();
inFile.close();
return 0;
}
```





# 总结

## 文件

- 文件的读写指针
- 字符文件读写，二进制文件读写
- 显式关闭文件
- 获得文件长度
- 命令行参数



# 附录：open和fopen的区别

## • 缓冲文件系统

- 在内存→缓冲区, 为程序中的每一个文件使用
- 当执行读文件的操作时, 从磁盘文件将数据→内存缓冲区
- 装满后再从内存缓冲区→接收的变量

内存缓冲区的大小, 影响着实际操作外存的次数, 内存缓冲区越大, 则操作外存的次数就少, 执行速度就快、效率高

## • 相关函数

fopen, fclose, fread, fwrite, fgetc, fgets, fputc, fputs,  
freopen, fseek, ftell, rewind



# 附录：open和fopen的区别

- 非缓冲文件系统

- 借助文件结构体指针来对文件进行管理
- 通过文件指针来对文件进行访问,既可以读写字符,字符串,格式化数据,也可以读写二进制数据
- 非缓冲文件系统依赖于操作系统,通过操作系统的功能对文件进行读写,是系统级的输入输出,它不设文件结构体指针,只能读写二进制文件,但效率高、速度快

- 相关函数:

- open, close, read, write, getc, getchar, putc, putchar

- open 是系统调用 返回的是文件句柄,文件的句柄是文件在文件描述副表里的索引

**Vs.** fopen是C的库函数,返回的是一个指向文件结构的指针

