§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.1. 流的基本概念

流的含义：流是来自设备或传给设备的一个数据流，由一系列字节组成，按顺序排列（也称为字节流）

★ C/C++的原生标准中没有定义输入/输出的基本语句

★ C语言用printf/scanf等函数来实现输入和输出，通过#include <stdio.h>来调用

★ C++通过cin和cout的流对象来实现，通过#include <iostream>来调用

cout：输出流对象　　<<：流插入运算符

cin：输入流对象　　>>：流提取运算符

§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.2. 输出流的基本操作

格式：cout << 表达式1 << 表达式2 << ... << 表达式n;

★ 插入的数据存储在缓冲区中，不是立即输出，要等到缓冲区满(不同系统大小不同)或者碰到换行符("\n"/endl)或者强制立即输出(flush)才一齐输出

★ 默认的输出设备是显示器(可更改，称输出重定向)

★ 一个cout语句可写为若干行，或者若干语句

★ 一个cout的输出可以是一行，也可以是多行，多个cout的输出也可以是一行

★ 一个插入运算符只能输出一个值

★ 系统会自动判断输出数据的格式

§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.3. 输入流的基本操作

格式：cin >> 变量1 >>变量2 >> ... >>变量n;

★ 键盘输入的数据存储在缓冲区中，不是立即被提取，要等到缓冲区满(不同系统大小不同)或碰到回车符才进行提取

★ 默认的输入设备是键盘(可更改，称输入重定向)

★ 一行输入内容可分为若干行，或者若干语句

★ 一个提取运算符只能输入一个值

★ 提取运算符后必须跟变量名,不能是常量/表达式等(因为提取数据后会改变 >> 后的值，因此 >> 后面必须是变量)

★ 输入终止条件为回车、空格、非法输入

★ 系统会自动根据cin后变量的类型按最长原则来读取合理数据

★ 变量读取后，系统会判断输入数据是否超过变量的范围，若超过则置内部的错误标记并返回一个不可信的值
(不同编译器处理不同)

★ cin输入完成后，通过cin自身/cin.good()/cin.fail()可判断本次输入是否正确

★ 输入时超数据范围，和赋值时超数据范围，系统的处理不相同

★ 字符型变量只能输入图形字符(33-126)，不能以转义符方式输入(单双引号、转义符全部当作单字符)

★ 浮点数输入时，可以是十进制数或指数形式，只取有效位数(4舍5入)

★ cin不能跟endl，否则编译错

§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.4. 在输入输出流中使用格式化控制符

C++缺省输入/输出格式是默认格式，为满足一些特殊要求，需要对数据进行格式化

§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.5. 字符的输入和输出

3.4.5.1. 字符输出函数putchar

形式：putchar(字符变量/常量)

功能：输出一个字符

```
char a='A';
putchar(a);
putchar('A') ;
putchar('\x41');
putchar('\101');
```

★ 加#include <cstdio>或#include <stdio.h>(目前VS/Dev均不需要)

★ 返回值是int型，是输出字符的ASCII码，可赋值给字符型/整型变量

§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.5. 字符的输入和输出

3.4.5.2. 字符输入函数getchar

形式：getchar()

功能：输入一个字符(给指定的变量)

★ 加#include <cstdio>或#include <stdio.h>(目前VS/Dev均不需要)

★ 返回值是int型，是输入字符的ASCII吗，可赋值给字符型/整型变量

★ 输入时有回显，输入后需按回车结束输入(若直接按回车则得到回车的ASCII码)

★ 可以输入空格，回车等cin无法处理的非图形字符，但仍不能处理转义符

★ cin/getchar 等每次仅从输入缓冲区中取需要的字节，多余的字节仍保留在输入缓冲区中供下次读取

§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.5. 字符的输入和输出

3.4.5.3. 字符输入函数_getch与_getche

★ 几个字符输入函数的差别

getchar ：有回显，不立即生效，需要回车键

_getche ：有回显，不需要回车键

_getch ：无回显，不需要回车键

★ 在Dev C++中

getch() ⇔ _getch()

getche() ⇔ _getche()

§8. 输入输出流

8.1. C++的输入与输出

8.1.1. 有关输入输出基本概念的回顾

§3. 结构化程序设计基础

3.4. C++的输入与输出

3.4.6. C语言的格式化输入与输出函数

★ 格式化输出：printf

★ 格式化输入：scanf

下发相关资料并参考其它书籍，结合作业进行自学

要求：

1、熟练使用C++的cin/cout

2、能看懂C的printf/scanf

3、除非明确规定，C++作业不允许printf/scanf

§8. 输入输出流

8.1. C++的输入与输出

8.1.2. 输入输出的基本概念

输入输出的种类：

系统设备：标准输入设备：键盘

(标准I/O) 标准输出设备：显示器

其它设备：鼠标、打印机、扫描仪等

外存文件：从文件中得到输入

(文件I/O) 输出到文件中

内存空间：输入/输出到一个字符数组/string中

(串I/O)

★ 操作系统将所有系统设备都统一当作文件进行处理

C++输入/输出的特点：

★ 与C兼容，支持printf/scanf

★ 对数据类型进行严格的检查，是类型安全的I/O操作

★ 具有良好的可扩展性，可通过重载操作符的方式输入/输出自定义数据类型(荣誉课：运算符重载)

C++的输入/输出流：

★ 采用字符流方式，缓冲区满或遇到endl才输入/输出

★ cin,cout不是C++的语句，也不是函数，是类的对象 >> 和 << 的本质是左移和右移运算符，被重载为输入和输出运算符(荣誉课：运算符重载)

§8. 输入输出流

8.2. 标准输出流

8.2.1. cout，cerr和clog流

cout：向控制台进行输出，缺省是显示器

cerr：向标准出错设备进行输出，缺省是显示器(直接输出，不必等待缓冲区满或回车)

clog：向标准出错设备进行输出，缺省是显示器(放在缓冲区中，等待缓冲区满或回车才输出)

★ 三者的使用方法一样

★ 缺省都是显示器，可根据需要进行输出重定向(视频补充内容)

8.2.2. 格式输出

需要掌握的基本格式：

不同数制：dec、hex、oct

设置宽度：setw

左右对齐：setiosflags(ios::left/right)

其余当作手册来查：

setfill、setprecision等

★ 输出格式可用控制符控制，也可以流成员函数形式

```
cout << setw(20)          ⇔ cout.width(20)
cout << setprecision(9)   ⇔ cout.precision(9)
...
```

§8. 输入输出流

8.2. 标准输出流

8.2.3. 流成员函数put

形式：cout.put(字符常量/字符变量)

★ 功能与putchar相同，输出一个字符

```
char a='A';
cout.put(a);        //变量
cout.put('A') ;     //常量
cout.put('\x41'); //十六进制转义符
cout.put('\101'); //八进制转义符
cout.put(65) ;      //整数当作ASCII码
cout.put(0x41) ;   //整数当作ASCII码(十六)
cout.put(0101) ;   //整数当作ASCII码(八)
```

★ 允许连续调用

```
#include <iostream>
using namespace std;

int main()
{
    cout.put(72).put(0x65).put('l').put(0154).put('a'+14);
    return 0;
}
```

```
Hello
```

§8. 输入输出流

8.3. 标准输入流

8.3.1. cin流

★ cin提取数据后，会根据数据类型是否符合要求而返回逻辑值

```
#include <iostream>
using namespace std;
int main()
{
    int a=-9;
    cin >> a;
    cout << a << " " << (cin ? 1 : 0) << endl;
    return 0;
} //不同编译器，cin为0时，a值可能不同
```

输入	cout的输出	
10	10	1
ab	0	0
12ab	12	1
很大的数字	2147483647	0

- 当cin返回为1/true时，读入的值才可信
  =>正确的处理逻辑：cin读入后，先判断cin，为1再取值
- 不同编译器，cin为0时，a的值可能不同(不可信)
- 也可以通过 cin.good() / cin.fail() 来判断

★ 允许进行输入重定向(视频补充内容)

§8. 输入输出流

8.3. 标准输入流

8.3.2. 文件结束符与文件结束标记

文件结束符：表示文件结束的特殊标记

★ 设备也当作文件处理

★ 一般用CTRL+Z表示键盘输入文件结束符(第05模块的PPT作业)

文件结束标记：判断文件是否结束的标记，用宏定义EOF来表示

★ 不同系统EOF的值可能不同，不必关心

★ 一般用于字符流输入的判断，对其它类型一般不用

8.3.3. 用于字符输入的流成员函数

★ cin.get()

★ cin.get(字符变量)

★ cin.get(字符数组，字符个数n，中止字符)

★ cin.getline(字符数组，字符个数n，中止字符)

★ cin.eof()

★ cin.peek()

★ cin.putback(字符变量/字符常量)

★ cin.ignore(字符个数n，中止字符)

作业

§8.输入输出流

8.4.文件操作与文件流

8.4.1.文件的基本概念

文件及文件名：

文件：存储在外存储器上的数据的集合

文件名：操作系统用于访问文件的依据

文件的分类：

★ 按设备分

输入文件：键盘等输入设备

输出文件：显示器、打印机等输出设备

磁盘文件：存放在磁盘(光盘、U盘)上的文件

★ 按文件的类型分：

程序文件：执行程序所对应的文件(.exe/.dll等)

数据文件：存放对应数据的文件(.cpp/.doc等)

★ 按数据的组织形式

ASCII码文件(文本文件)：按数据的ASCII代码形式存放的文件

二进制文件：按数据的内存存放形式存放的文件

§8. 输入输出流

8.4. 文件操作与文件流

8.4.1. 文件的基本概念

文件的分类：

★ 按数据的组织形式

ASCII码文件(文本文件)：按数据的ASCII代码形式存放的文件

二进制文件：按数据的内存存放形式存放的文件

例：int型整数100000：ASCII文件为6个字节

二进制文件为4个字节

\x31	\x30	\x30	\x30	\x30	\x30

\x00	\x01	\x86	\xA0

100000=0x186A0

双精度数123.45：ASCII文件为6个字节

二进制文件为8个字节

\x31	\x32	\x33	\x2E	\x34	\x35

IEEE754 作业

d	o	u	b	l	e	格	式

字符串"China"：ASCII文件为5个字节

二进制文件为6个字节

\x43	\x68	\x69	\x6E	\x61

\x43	\x68	\x69	\x6E	\x61	\x0

§8. 输入输出流

8.4. 文件操作与文件流

8.4.1. 文件的基本概念

C++对文件的访问：

低级I/O：字符流方式输入/输出

高级I/O：转换为数据指定形式的输入/输出

8.4.2. 文件流类及文件流对象

与磁盘文件有关的流类：

输入　　：ifstream类，从istream类派生而来

输出　　：ofstream类，从ostream类派生而来

输入/输出：fstream类，从iostream类派生而来

流对象的建立：

ifstream 流对象名 ：用于输入文件的操作

ofstream 流对象名 ：用于输出文件的操作

fstream 流对象名 ：用于输入/输出文件的操作

§8. 输入输出流

8.4. 文件操作与文件流

8.4.3. 文件的打开与关闭

文件的打开：

文件流对象名.open(文件名，打开方式);

★ 加#include <fstream>

★ 有多种打开方式

ios::nocreate
ios::noreplace

DevC++/Linux不支持

在VS下是

ios::_Nocreate
ios::_Noreplace

方　式	作　用
ios :: in	以输入方式打开文件
ios :: out	以输出方式打开文件(这是默认方式),如果已有此名字的文件,则将其原有内容全部清除
ios :: app	以输出方式打开文件,写入的数据添加在文件末尾
ios :: ate	打开一个已有的文件,文件指针指向文件末尾
ios :: trunc	打开一个文件,如果文件已存在,则删除其中全部数据;如文件不存在,则建立新文件。如已指定了 ios :: out 方式,而未指定 ios :: app,ios :: ate,ios :: in,则同时默认此方式
ios :: binary	以二进制方式打开一个文件,如不指定此方式则默认为 ASCII 方式
ios :: nocreate	打开一个已有的文件,如文件不存在,则打开失败。nocreat 的意思是不建立新文件
ios :: noreplace	如果文件不存在则建立新文件,如果文件已存在则操作失败,noreplace 的意思是不更新原有文件
ios :: in | ios :: out	以输入和输出方式打开文件,文件可读可写
ios :: out | ios :: binary	以二进制方式打开一个输出文件
ios :: in | ios :: binary	以二进制方式打开一个输入文件

★ 各个打开方式可用"位或运算符 |"进行组合

§8. 输入输出流

8.4. 文件操作与文件流

8.4.3. 文件的打开与关闭

文件的打开：

文件流对象名.open(文件名，打开方式);

1、路径有绝对路径和相对路径两种
2、..表示父目录，.表示当前目录
3、为什么要\\而不能\ ？
4、假设程序在D:\test下运行，这5个open，分别打开的是下面的哪个aa.dat文件？

C:\
|--test （文件夹）
| |--aa.dat
| |--C++ （文件夹）
| |--aa.dat
|--C++ （文件夹）
|--aa.dat

D:\
|--test （文件夹）
| |--aa.dat
| |--C++ （文件夹）
| |--aa.dat
|--C++ （文件夹）
|--aa.dat

★ 文件名允许带全路径，若不带路径，则表示与可执行文件同目录

```
ofstream out;
out.open("aa.dat", ios::out);
out.open("..\\C++\\aa.dat", ios::out);
out.open(".\\C++\\aa.dat", ios::out);
out.open("\\C++\\aa.dat", ios::out | ios::app);
out.open("c:\\C++\\aa.dat", ios::out);
```

● VS等编译器，如在集成环境内运行，则当前目录是指源程序文件(*.cpp)所在的目录，如果离开集成环境(例如用cmd命令行运行)，则当前目录是指可执行文件(*.exe)所在目录

注意：当前目录

★ 用"\\"表示目录间分隔符的方式在Linux下无效，用"/"方式则在Windows/Linux下均有效

```
out.open("aa.dat", ios::out);
out.open("../C++/aa.dat", ios::out);
out.open("./C++/aa.dat", ios::out);
out.open("/C++/aa.dat", ios::out | ios::app);
out.open("c:/C++/aa.dat", ios::out);
```

§8. 输入输出流

8.4. 文件操作与文件流

8.4.3. 文件的打开与关闭

文件的打开：

文件流对象名.open(文件名，打开方式);

★ 可在声明文件流对象时直接打开

```
ofstream out("aa.dat", ios::out);
```

★ 打开方式与文件流对象之间要兼容,否则无意义

```
ifstream in;
in.open("aa.dat", ios::out);    //in对象用out打开，无意义，缺省仍为 ios::in，具体见后例
```

★ 每个文件被打开后，都有一个文件指针，初始指向开始/末尾的位置

- 根据打开方式决定
- 用于指示在当前文件中的偏移位置（与指针变量的概念无关）
- 细节见作业

★ 执行open操作后，要判断文件是否打开成功

```
if (!outfile)                                  两种方法均可以
if (outfile.is_open()==0)                      1、判断流对象自身
if (!outfile.is_open())                        2、is_open函数
```

```
if (outfile.open("f1.dat",ios::app)==0)  //open返回void
if (!outfile.open("f1.dat",ios::app))                    错
if (outfile==NULL)                       //outfile不是指针
```

文件的关闭：

文件流对象名.close();

§8. 输入输出流

8.4. 文件操作与文件流

8.4.3. 文件的打开与关闭

文件的打开：

文件流对象名.open(文件名，打开方式);

文件的关闭：

文件流对象名.close();

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{   ifstream in;
    in.open("bb.dat", ios::in);
    if (in.is_open()==0)
        cout << "open failed." << endl;
    else
        cout << "open success." << endl;
    in.close();

    return 0;
}
```

不存在时：失败
存在时：成功

若换成ios::out，则打开不受影响，但后续读写会有问题

ifstream

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{   ofstream out;
    out.open("aa.dat", ios::out);
    if (out.is_open()==0)
        cout << "open failed." << endl;
    else
        cout << "open success." << endl;
    out.close();

    return 0;
}
```

不存在时：成功(创建)
存在时：成功(覆盖)
存在并只读：失败

ofstream

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{   ofstream out;
    out.open("bb.dat", ios::out | ios::_Nocreate);
    if (out.is_open()==0)
        cout << "open failed." << endl;
    else
        cout << "open success." << endl;
    out.close();
}
```

不存在时：失败
存在时：成功(覆盖)
存在并只读：失败

ofstream

§8. 输入输出流

8.4. 文件操作与文件流

8.4.4. 对ASCII文件的操作

基本方法：将文件流对象名当作cin/cout对象，用 >> 和 << 进行格式化的输入和输出，同时前面介绍的关于cin/cout的get/getline/put/eof/peek/putback/ignore等成员函数也可以被文件流对象所使用

★ >>和<<使用时的注意事项与cin、cout时相同

cin >> 变量 => infile >> 变量

cout << 变量 => outfile << 变量

★ 成员函数的使用方法与前面相同

cout.put('A') => outfile.put('A')

★ 流对象与打开方式、流插入/流提取运算符之间要求匹配

● 错误的例子：ifstream打开的写文件用流插入运算符

```
//例：打开d:\test\data.txt文件,并写入"Hello"
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream out;
    char ch;
    out.open("d:/test/data.txt", ios::out); //ios::out无效
    if (out.is_open()==0) {
        cout << "文件打开失败" << endl;
        return -1;
        }
    out << "Hello" << endl; //编译错!!!
    out.close();
    return 0;
}
```

error C2676: 二进制“<<”：“std::ifstream”不定义该运算符或到预定义运算符可接收的类型的转换

§8. 输入输出流

8.4. 文件操作与文件流

8.4.4. 对ASCII文件的操作

```
//例：键盘输入10个int，输出到文件中
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
   int a[10];
   ofstream outfile("f1.dat", ios::out);
   if (!outfile.is_open()) {
      cout << "文件打开失败" << endl;
      return -1;
      }
   cout << "enter 10 integer numbers:" << endl;
   for(int i=0; i<10; i++) {
      cin >> a[i];                  //键盘输入
      outfile << a[i] << " ";       //int型输出到文件
      }
   outfile.close();
   return 0;
}
```

运行两次，观察结果

```
//例：键盘输入10个int，输出到文件中（变化，加ios::app）
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
   int a[10];
   ofstream outfile("f1.dat", ios::out | ios::app);
   if (!outfile.is_open()) {
      cout << "文件打开失败" << endl;
      return -1;
      }
   cout << "enter 10 integer numbers:" << endl;
   for(int i=0; i<10; i++) {
      cin >> a[i];                  //键盘输入
      outfile << a[i] << " ";       //int型输出到文件
      }
   outfile.close();
   return 0;
}
```

运行两次，观察结果

§8. 输入输出流

8.4. 文件操作与文件流

8.4.4. 对ASCII文件的操作

```
//例：从文件中读入10个int，输出到屏幕上
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    int a[10];
    ifstream infile("f1.dat", ios::in);
    if (!infile.is_open()) {
        cout << "文件打开失败" << endl;
        return -1;
        }
    for(int i=0; i<10; i++) {
        infile >> a[i];          //从文件中读10个int放入a数组
        cout << a[i] << " ";     //int型输出到屏幕
        }

   infile.close();
   return 0;
}
```

利用上例生成的f1.dat
自己编辑完全正确的f1.dat
自己编辑含错误的f1.dat

§8. 输入输出流

8.4. 文件操作与文件流

8.4.4. 对ASCII文件的操作

```
//例：打开d:\test\data.txt文件,并将内容输出到屏幕上
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream in;
    char ch;
                "d:/test/data.txt"
    in.open("d:\\test\\data.txt", ios::in);//双斜杠
    if (in.is_open()==0) {  //!in.is_open()
        cout << "文件打开失败" << endl;
        return -1;
        }

    while(!in.eof()) {
        ch = in.get();    //in.get(ch);
        putchar(ch);      //cout.put(ch);
        }

    in.close();
    return 0;
}
```

```
while((ch=in.get())!=EOF)
    cout.put(ch);

EOF是系统定义的文件结束标记
```

```
//例：将 d:\test\data.txt 文件复制为 d:\demo\data2.txt
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream in;
    ofstream out;
    char ch;
    in.open("d:\\test\\data.txt", ios::in);
    if (!in.is_open()) {
        cout << "无法打开源文件" << endl;
        return -1;
        }
    out.open("d:\\demo\\data2.txt", ios::out);
    if (!out.is_open()) {
        cout << "无法打开目标文件" << endl;
        in.close();//记得关掉
        return -1;
        }

    while(in.get(ch))
        out.put(ch);

    in.close();
    out.close();
    return 0;
}
```

"d:/test/data.txt"

```
while(!in.eof()) {
    ch = in.get();
    out.put(ch);
    }
```

问：1、左右两种复制方式，哪种方式复制后的字节大小与原文件不同？为什么？
2、在保持读源文件的函数不变的情况下如何改正？
3、左侧的输出到屏幕有同样问题吗？

§8. 输入输出流

8.4. 文件操作与文件流

8.4.5. 对二进制文件的操作

★ 用ASCII文件的字符方式进行操作

- 仅能按字节读写
- 如果文件中有0x1A则无法继续读取
  - 0x1A = 26（ASCII码值26） => CTRL+Z => EOF
  - 正常文本文件不可能有此字符

★ 用read/write进行操作

文件流对象名.read(内存空间首指针，长度);

从文件中读长度个字节，放入从首指针开始的空间中

文件流对象名.write(内存空间首指针，长度);

将从首指针开始的连续长度个字节写入文件中

- read/write均为纯字节，无尾零等任何附加信息
- read/write一般仅用于二进制读写，如果用于十进制读写，则仅受长度的限制，不考虑格式化

(是否有尾零/数据是否合理等，相当于二进制)

§8. 输入输出流

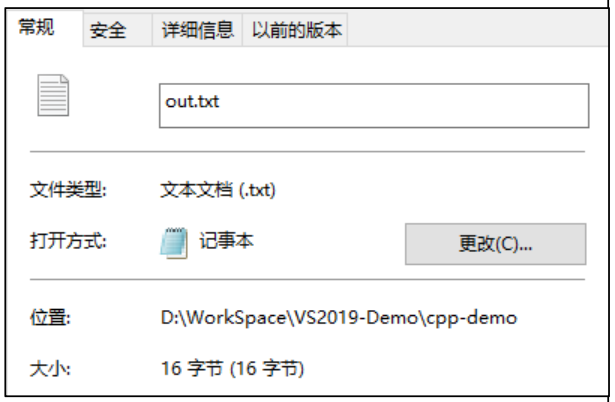
8.4. 文件操作与文件流

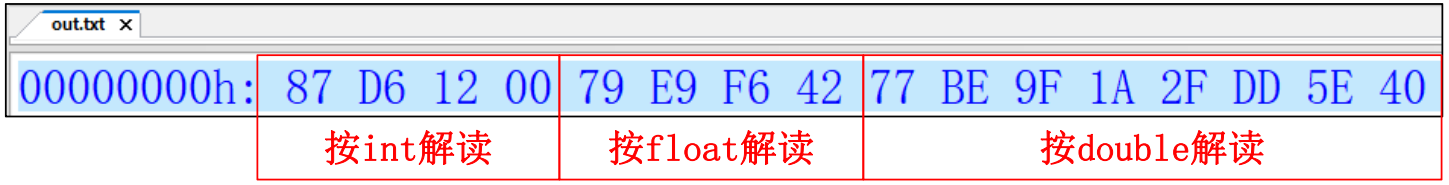
8.4.5. 对二进制文件的操作

★ 用read/write进行操作

```
//例：将内存以原始二进制方式写入文件中
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream out("out.txt", ios::out);//再加 ios::binary
    int    i = 1234567;               //有区别吗？
    float  f = 123.456f;
    double d = 123.456;

    out.write((char *)&i, sizeof(int));
    out.write((char *)&f, sizeof(float));
    out.write((char *)&d, sizeof(double));

    out.close();
    return 0;
}
```

以ASCII文件方式写入





§8. 输入输出流

8.4. 文件操作与文件流

8.4.5. 对二进制文件的操作

★ 用read/write进行操作

```
//例：将内存以ASCII方式写入文件中（二进制转十进制）
#include <iostream>
#include <fstream>
using namespace std;

struct student {
   char name[20];
   int  num;
   int  age;
   char sex;
}; //含填充共32字节

int main()
{
    student stud[3]={"Li",1001,18,'f', "Fan",1002,19,'m',"Wang",1004,17,'f'}; //每个数组的存储为机内二进制形式
    ofstream outfile("stud.dat", ios::out);
    if (!outfile.is_open()) {
        cout << "文件打开失败" << endl;
        return -1;
        }

    for(int i=0; i<3; i++)
        outfile << stud[i].name << stud[i].num  << stud[i].age << stud[i].sex << endl; //通过 << 转换为十进制形式

    outfile.close();
    return 0;
}
```

以ASCII文件方式写入

生成的stud.dat文件共36字节：

Li100118f
Fan100219m
Wang100417f

8.4. 文件操作与文件流

8.4.5. 对二进制文件的操作

★ 用read/write进行操作

stud.dat

```
00000000h: 4C 69 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000010h: 00 00 00 00 E9 03 00 00 12 00 00 00 66 CC CC CC
00000020h: 46 61 6E 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030h: 00 00 00 00 EA 03 00 00 13 00 00 00 6D CC CC CC
00000040h: 57 61 6E 67 00 00 00 00 00 00 00 00 00 00 00 00
00000050h: 00 00 00 00 EC 03 00 00 11 00 00 00 66 CC CC CC
```

stud.dat文件共96字节，前32字节为：

```
4C 69 00 ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??
?? ?? ?? ?? E9 03 00 00 12 00 00 00 66 ?? ?? ??
```

4C6900 => "Li"(含尾零，多余17个)

E9030000 => 0x000003E9（低位在前） => 1001

12000000 => 0x00000012 => 18

66 => 'f' （多余3个CC是填充字节）

以二进制文件方式写入


```
//例：将内存以原始二进制方式写入文件中
#include <iostream>
#include <fstream>
using namespace std;

struct student {
   char name[20];
   int  num;
   int  age;
   char sex;
}; //含填充共32字节

int main()
{
    student stud[3]={"Li",1001,18,'f', "Fun",1002,19,'m',"Wang",1004,17,'f'}; //每个数组的存储为机内二进制形式
    ofstream outfile("stud.dat", ios::binary);
    if (!outfile.is_open()) {
        cout << "文件打开失败" << endl;
        return -1;
        }

    for(int i=0; i<3; i++) //一次写一个数组元素（32字节）
        outfile.write((char *)&stud[i], sizeof(stud[i]));

    outfile.close();
    return 0;
}
```

outfile.write((char *)stud, sizeof(stud));//整个数组96字节）

§8. 输入输出流

8.4. 文件操作与文件流

8.4.5. 对二进制文件的操作

★ 用read/write进行操作

以二进制文件方式读取

```
//用read方式读二进制文件并以十进制方式输出到屏幕上（stud.dat由上例生成）
#include <iostream>
#include <fstream>
using namespace std;
struct student {
   char name[20];  //不能是string name，必须是char name[20]，[19]或[21]都不行，必须要保证与文件的32字节一致
   int  num;
   int  age;
   char sex;
}; //含填充共32字节
int main()
{   student stud[3];
    int i;
    ifstream infile("stud.dat", ios::binary); //stud.dat的内容是由上例生成的二进制文件
    if (!infile.is_open()) {
         cout << "文件打开失败" << endl;
         return -1;
         }
    for(i=0; i<3; i++)  //一次读入一个数组元素（32字节）
         infile.read((char *)&stud[i], sizeof(stud[i]));
    infile.close();
    for(i=0; i<3; i++) {
         cout << "No." << i+1 << endl;
         cout << "name:" << stud[i].name << endl;
         cout << "num:" << stud[i].num << endl;
         cout << "age:" << stud[i].age << endl;
         cout << "sex:" << stud[i].sex << endl << endl; //多空一行
         }
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
No.1
name:Li
num:1001
age:18
sex:f

No.2
name:Fun
num:1002
age:19
sex:m

No.3
name:Wang
num:1004
age:17
sex:f
```

stud.dat

```
00000000h: 4C 69 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000010h: 00 00 00 00 E9 03 00 00 12 00 00 00 66 CC CC CC
00000020h: 46 61 6E 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030h: 00 00 00 00 EA 03 00 00 13 00 00 00 6D CC CC CC
00000040h: 57 61 6E 67 00 00 00 00 00 00 00 00 00 00 00 00
00000050h: 00 00 00 00 EC 03 00 00 11 00 00 00 66 CC CC CC
```

§8. 输入输出流

8.4. 文件操作与文件流

8.4.5. 对二进制文件的操作

★ 与文件指针有关的流成员函数

适用于输入文件的：

gcount()　　：返回最后一次读入的字节

tellg()　　：返回输入文件的当前指针

seekg(位移量，位移方式)：移动输入文件指针

适用于输出文件的：

tellp()　　：返回输出文件的当前指针

seekp(位移量，位移方式)：移动输出文件指针

位移方式：

ios::beg：从文件头部移动，位移量必须为正

ios::cur：从当前指针处移动，位移量可正可负

ios::end：从文件尾部移动，位移量必须为负

★ 随机访问二进制数据文件

在文件的读写过程中，可前后移动文件指针，达到按需读写的目的

- ifstream无tellp，ofstream无tellg
- fstream的tellg/tellp是同步移动的

细节通过作业理解

★ 关于二进制访问的几个注意事项

- read/write虽然是内存首地址，实际编程中用字符数组，但注意不是字符串，不处理\0
- read参数中的长度是最大读取长度，不是实际读取长度，因此read后要用gcount()返回真实读到的字节数
- 如果读写方式打开(ios::in | ios::out)，则只有一个文件指针，seekg()和seekp()是同步的，tellg()和tellp()也是同步的
- 在文件的操作超出正常范围后(例：read()已到EOF、seekg()/seekp()超文件首尾等)，再次对文件进行seekg()/seekp()/tellg()/tellp()等操作都可能会返回与期望不同的值，建议在文件操作过程中多用good()/fail()/eof()/clear()等函数，具体自行体会

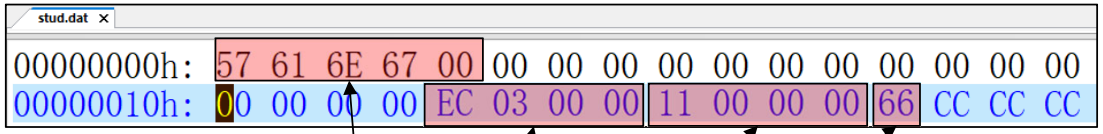
§8. 输入输出流

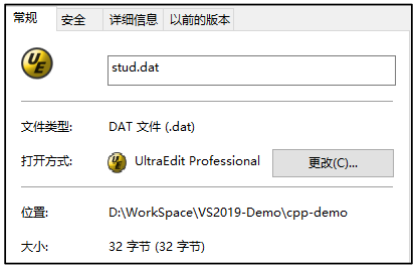
8.4. 文件操作与文件流

8.4.5. 对二进制文件的操作

★ 用read/write进行操作+文件指针移动

```
#include <iostream>
#include <fstream>
using namespace std;

struct student {
   char name[20];
   int  num;
   int  age;
   char sex;
}; //含填充共32字节

int main()
{
    student stud[3]={"Li",1001,18,'f',"Fan",1002,19,'m',"Wang",1004,17,'f'};
    ofstream outfile("stud.dat", ios::binary);
    if (!outfile.is_open()) {
        cout << "文件打开失败" << endl;
        return -1;
        }
    for(int i=0; i<3; i++) { //一次写入一个数组元素
        outfile.write((char *)&stud[i],sizeof(stud[i]));//从当前位置写32字节
        cout << outfile.tellp() << endl;  //写完32字节后，文件指针为32
        outfile.seekp(0, ios::beg);       //文件指针移回0
        cout << outfile.tellp() << endl;  //文件指针为0
        }
    outfile.close();
    return 0;
}
```





1、初始文件指针为0，写32字节
2、随后移回0
3、再次写32字节时会覆盖上次的32字节
4、最终文件大小为32字节

§8. 输入输出流

8.4. 文件操作与文件流

8.4.5. 对二进制文件的操作

★ 用read/write进行操作

综合应用：读写同时进行

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
using namespace std;
struct student {  //结构体与前面不同，28字节
   int num;
   char name[20];
   float score;
};
int main()
{   student stud[5]={1001,"Li",85, 1002, "Fan",97.5, 1004, "Wang",54, 1006, "Tan",76.5, 1010, "Ling", 96 };
    fstream iofile("stud.dat", ios::in | ios::out | ios::binary); //二进制打开，同时I/O
    if (!iofile.is_open()) {
       cout << "文件打开失败" << endl;
       return -1;  //强制结束程序的运行
       }
    /* 将stud[5]写入stud.dat中，大小为28*5=140 */
    for(int i=0; i<5; i++)
       iofile.write((char *)&stud[i], sizeof(stud[i]));

    student stud1[5];
    for(int i=0; i<5; i+=2) {
       /* i=0/2/4，偏移量为（0-27、56-83、112-140字节）*/
       iofile.seekg(i*sizeof(stud[i]), ios::beg);
       /* 将文件的第0/2/4个学生的信息放入stud[5]的0/1/2中 */
       iofile.read((char *)&stud1[i/2], sizeof(stud1[0]));
       cout << stud1[i/2].num << stud1[i/2].name << endl;//屏幕输出
       }
    cout << endl;

    stud[2].num = 1012; //修改第[2]个信息
    strcpy(stud[2].name, "Wu");
    stud[2].score = 60;

    /* 定位在第56字节，覆盖了文件中第2个学生的信息 */
    iofile.seekp(2*sizeof(stud[0]), ios::beg);
    iofile.write((char *)&stud[2], sizeof(stud[2]));//覆盖[2]
    //文件指针移动到最开始
    iofile.seekg(0, ios::beg);
    for(int i=0; i<5; i++) { //循环，在屏幕上输出文件中的内容
       iofile.read((char *)&stud[i], sizeof(stud[i]));
       cout << stud[i].num << stud[i].name << stud[i].score << endl;
       }
    iofile.close();
    return 0;
}
```

```
1001Li
1004Wang
1010Ling
```

```
1001Li85
1002Fan97.5
1012Wu60
1006Tan76.5
1010Ling96
```

Microsoft Visual Studio 调试控制台

```
1001Li
1004Wang
1010Ling

1001Li85
1002Fan97.5
1012Wu60
1006Tan76.5
1010Ling96
```