§17. 输入输出流

要求：

1、安装UltraEdit软件，学会使用16进制方式查看文件，并掌握ASCII及16进制查看间的切换

2、完成本文档中所有的测试程序并填写运行结果，从而体会二进制与十进制文件在不同操作系统下的读写差异，掌握与文件有关的流函数的正确用法

3、需完成的页面，右上角有标注，直接在本文件上作答，用蓝色写出答案/截图即可；填写答案时，为适应所填内容或贴图，允许调整页面的字体大小、文本框的位置等

4、转换为pdf后提交

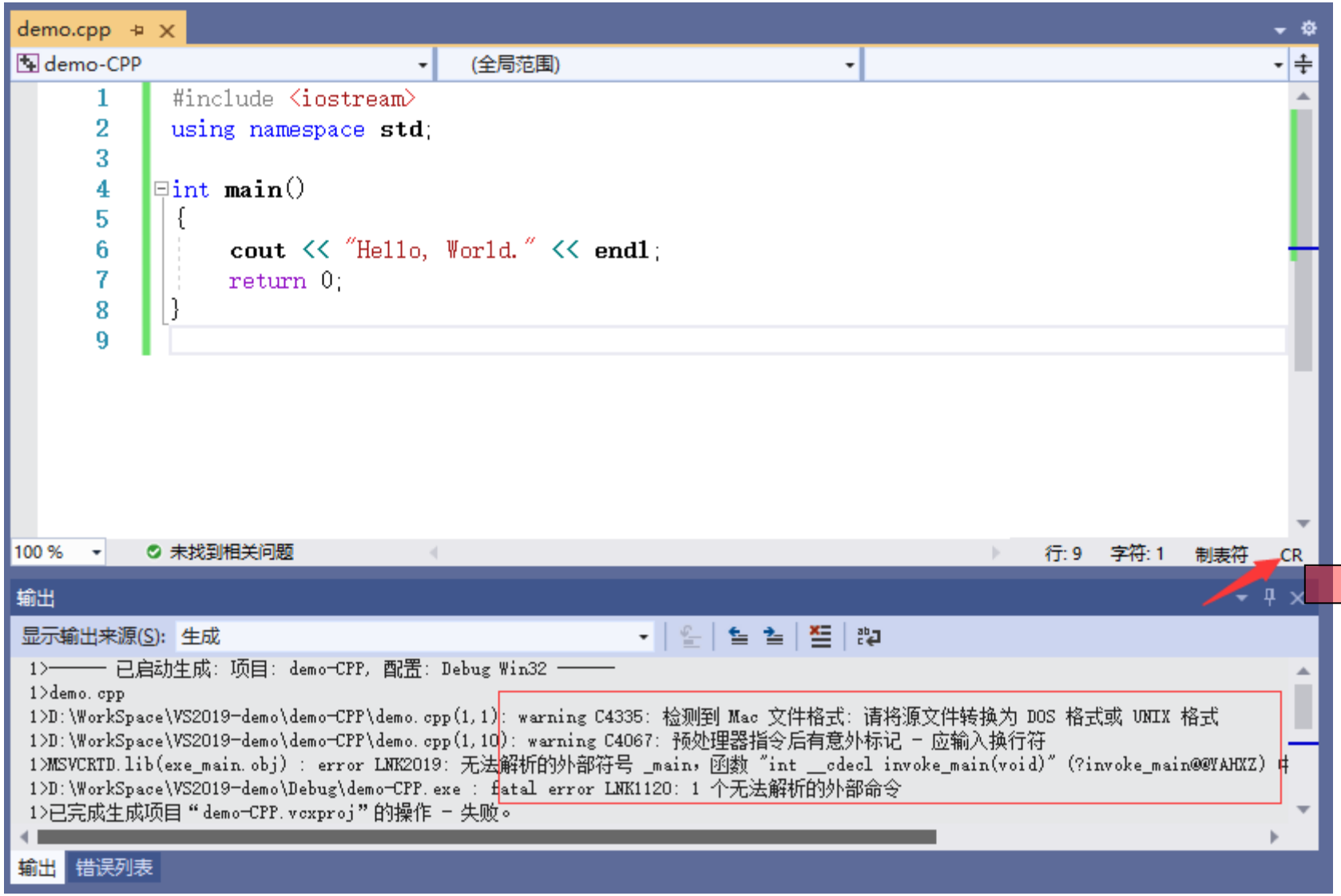
5、无特殊说明，Windows下用VS2019编译，Linux下用C++编译

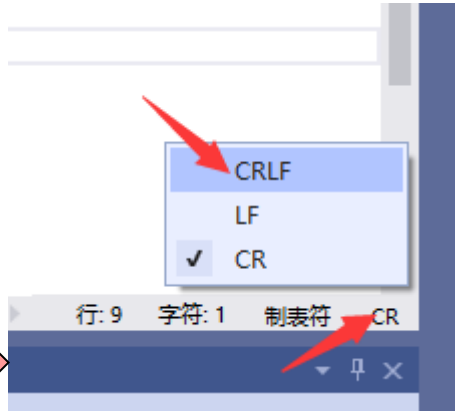
6、因为篇幅问题，打开文件后均省略了是否打开成功的判断，这在实际应用中是不允许的

7、5月27日前网上提交本次作业（在“实验报告”中提交）

§17. 输入输出流

附：用WPS等其他第三方软件打开PPT，将代码复制到VS2019中后，如果出现类似下面的编译报错，则观察源程序编辑窗的右下角是否为CR，如果是，单击CR，在弹出中选择CRLF，再次CTRL+F5运行即可





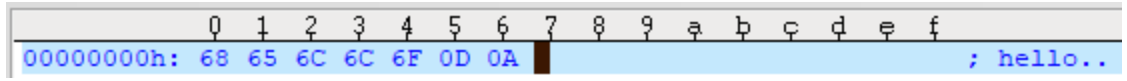
§17. 输入输出流

本页需填写答案

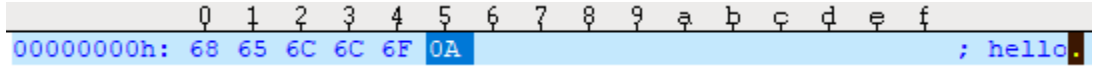
例1：十进制方式写，在Windows/Linux下的差别

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);

    out << "hello" << endl;

    out.close();

    return 0;
}
```

Windows下运行，out.txt是__7___字节，用UltraEdit的16进制方式打开的贴图



Linux下运行，out.txt是__6___字节，用UltraEdit的16进制方式打开的贴图



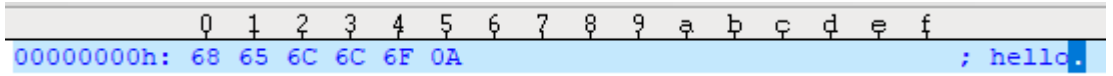
§17. 输入输出流

本页需填写答案

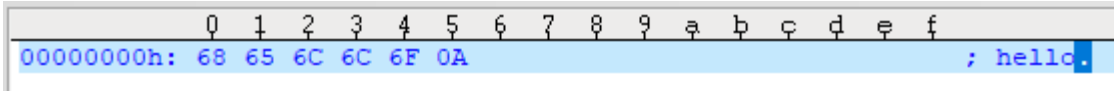
例2：二进制方式写，在Windows/Linux下的差别

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);

    out << "hello" << endl;

    out.close();

    return 0;
}
```

Windows下运行，out.txt是__6___字节，用UltraEdit的16进制方式打开的贴图



Linux下运行，out.txt是__6___字节，用UltraEdit的16进制方式打开的贴图



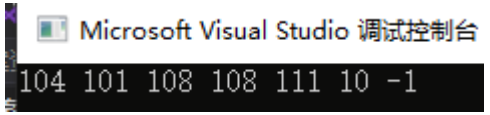
§17. 输入输出流

本页需填写答案

例3：十进制方式写，十进制方式读，0D0A(即"\r\n")在Windows下的表现

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    ifstream in("out.txt", ios::in);
    while(!in.eof())
        cout << in.get() << ' ';
    cout << endl;
    in.close();
    return 0;
}
```

Windows下运行，输出结果是：



说明：0D 0A在Windows的十进制方式下被当做__1__个字符处理，值是__10___。

§17. 输入输出流

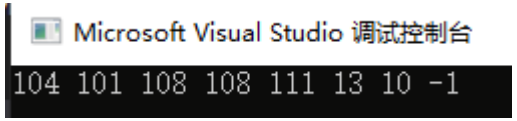
本页需填写答案

例4：十进制方式写，二进制方式读，0D0A(即"\r\n")在Windows下的表现

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    while(!in.eof())
        cout << in.get() << ' ';
    cout << endl;
    in.close();
    return 0;
}
```

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 68 65 6C 6C 6F 0D 0A                        ; hello..
```

Windows下运行，输出结果是：



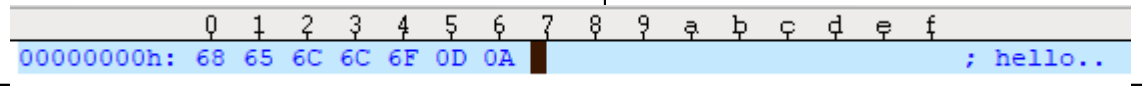
说明：0D 0A在Windows的二进制方式下被当做_2_个字符处理，值是_13 10_____。

§17. 输入输出流

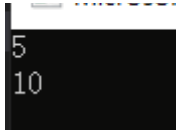
本页需填写答案

例5：十进制方式写，十进制方式读，不同读方式在Windows下的表现

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```



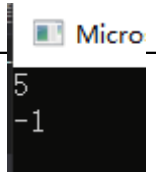
Windows下运行，输出结果是：



说明：in>>str读到_\n_就结束了，_\n__还被留在缓冲区中，因此in.peek()读到了 \n。

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

Windows下运行，输出结果是：



说明：in.getline读到__\n_就结束了，_\n被读掉，因此in.peek()读到了_文件结束__。

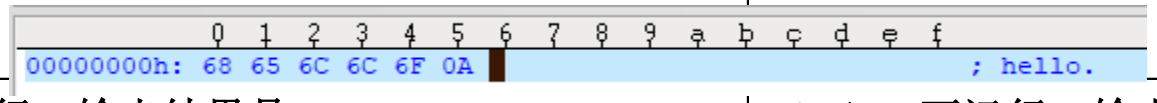
§17. 输入输出流

本页需填写答案

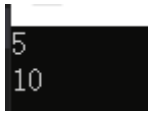
例6：二进制方式写，十进制方式读，不同读方式在Windows下的表现

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```



Windows下运行，输出结果是：



说明：in>>str读到_0A_就结束了，_0A_还被留在缓冲区中，因此in.peek()读到了_0A__。

Windows下运行，输出结果是：

说明：in.getline读到_0A_就结束了，_0A_被读掉，因此in.peek()读到了__文件结束_。

§17. 输入输出流

本页需填写答案

例7：二进制方式写，二进制方式读，不同读方式在Windows下的表现

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out | ios::binary);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 68 65 6C 6C 6F 0A                     ; hello.
```

Windows下运行，输出结果是：

```
5
10
```

说明：in>>str读到_0A_就结束了，_0A_还被留在缓冲区中，因此in.peek()读到了_0A__。

Windows下运行，输出结果是：

```
5
-1
```

说明：in.getline读到_0A__就结束了，_0A__被读掉，因此in.peek()读到了_文件结束__。

§17. 输入输出流

本页需填写答案

例8：十进制方式写，二进制方式读，不同读方式在Windows下的表现

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in >> str;
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 68 65 6C 6C 6F 0D 0A                    ; hello..
```

Windows下运行，输出结果是：

```
5
13
```

说明：in>>str读到__0D0A_就结束了，_0D0A_还被留在缓冲区中，因此in.peek()读到了_0D__。

Windows下运行，输出结果是：

```
6
-1
```

说明：

1、in.getline读到__0D0A_就结束了，_0D0A_被读掉，因此in.peek()读到了_文件结束_。

2、strlen(str)是_6__，最后一个字符是_0D

§17. 输入输出流

本页需填写答案

例9：在Linux读取Wwindows下写的十进制文件

在Linux下运行本程序

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello\r" << endl; //模拟Windows格式
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

```
          0  1  2  3  4  5  6
00000000h: 68 65 6C 6C 6F 0D 0A
```

Linux下运行，输出结果是：

```
6
-1
```

说明：

1、in.getline读到_0A_就结束了，_0A_被读掉，因此in.peek()读到了_文件结尾__。

2、strlen(str)是_6_，最后一个字符是_0D

同例8右侧，未变过

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello" << endl;
    out.close();

    char str[80];
    ifstream in("out.txt", ios::in | ios::binary);
    in.getline(str, 80);
    cout << strlen(str) << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

```
          0  1  2  3  4  5
00000000h: 68 65 6C 6C 6F 0A
```

本例说明，在Linux下读取Windows格式的文件，要注意0D的处理

Linux下运行，输出结果是：

```
5
-1
```

说明：

1、in.getline读到_0A_就结束了，_0A_被读掉，因此in.peek()读到了_文件结尾__。

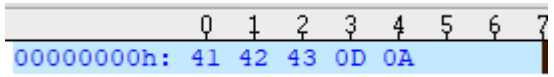
2、strlen(str)是_5__，最后一个字符是_6F_

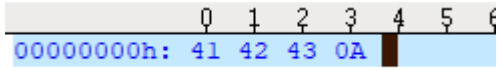
§17. 输入输出流

本页需填写答案

例10：用十进制方式写入含\0的文件，观察文件长度

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\0\x61\x62\x63" << endl;
    out.close();

    return 0;
}
```





Windows下运行，out.txt的大小是_5__字节，Linux下运行，out.txt的大小是__4__字节

为什么？

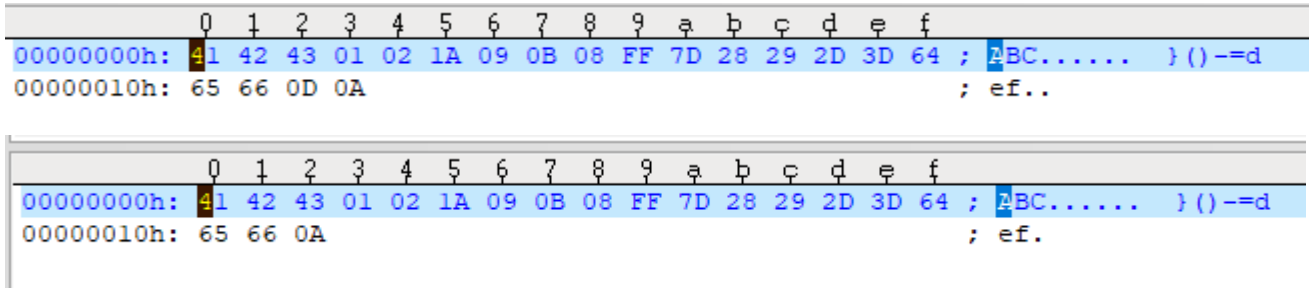
cout 前面那个字符串的时候，输出完ABC之后就是\0，停止了，就是 41 42 43
Windows下 endl 是 0D0A，加上前面三个，总共5个。而Linux下 endl 是0A，所以总共四个

§17. 输入输出流

本页需填写答案

例11：用十进制方式写入含非图形字符(ASCII码32是空格，33-126为图形字符)，但不含\0

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\xff\175()-=def" << endl;
    out.close();

    return 0;
}
```



Windows下运行，out.txt的大小是_20__字节，UltraEdit的16进制显示截图为：

Linux下运行，out.txt的大小是_19_字节，UltraEdit的16进制显示截图为：

§17. 输入输出流

本页需填写答案

例12：用十进制方式写入含\x1A(十进制26=CTRL+Z)的文件，并用十进制/二进制方式读取

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\xff\175()-=def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in);
    int c=0;
    while(!in.eof()) {
        in.get();
        c++;
        }
    cout << c << endl;
    in.close();

    return 0;
}
```

```
0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64
65 66 0D 0A
```

```
0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64
65 66 0A
```

Windows下运行，文件大小：____20_____
输出的c是：___6_____
Linux下运行，文件大小：____19_____
输出的c是：___20_____
为什么？\x1A 是win下十进制的结束输入符，c++那里是算上最后一次失败的，所以会比大小多1

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\xff\175()-=def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    int c=0;
    while(!in.eof()) {
        in.get();
        c++;
        }
    cout << c << endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：____20_____
输出的c是：___21_____
Linux下运行，文件大小：____19_____
输出的c是：___20_____
c的大小比文件大小大_1_，原因是：c++ 那里是算上最后一次失败的，所以会比大小多1

§17. 输入输出流

本页需填写答案

例13：用十进制方式写入含\x1A(十进制26=CTRL+Z)的文件，并用十进制不同方式读取

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\175()-=def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in);//不加ios::binary
    int c=0;
    while(in.get()!=EOF) {
        c++;
        }
    cout << c << endl;
    in.close();

    return 0;
}
```

```
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
41 42 43 01 02 1A 09 0B 08 7D 28 29 2D 3D 64 65
66 0D 0A
```

```
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
41 42 43 01 02 1A 09 0B 08 7D 28 29 2D 3D 64 65
66 0A
```

Windows下运行，文件大小：___19_____
输出的c是：___5_____
Linux下运行，文件大小：____18_____
输出的c是：___18_____
为什么？\x1A 是win下十进制的结束输入符，c++那里是先判断，没结束再加，所以和读到大小相同

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\x1A\t\v\b\175()-=def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in); //不加ios::binary
    int c=0;
    char ch;
    while((ch=in.get())!=EOF) {
        c++;
        }
    cout << c << endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：____19_____
输出的c是：___5_____
Linux下运行，文件大小：_____18____
输出的c是：____18____
为什么？Win 中十进制 in.get()读到 \x1A，返回值也是EOF

§17. 输入输出流

本页需填写答案

例14：用十进制方式写入含\xFF(十进制255/-1，EOF的定义是-1)的文件，并进行正确/错误读取

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\xff\t\v\b\175()-=def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in);//可加ios::binary
    int c=0;
    while(in.get()!=EOF) {
        c++;
        }
    cout << c << endl;
    in.close();

    return 0;
}
```

```
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
41 42 43 01 02 FF 09 0B 08 7D 28 29 2D 3D 64 65
66 0D 0A
```

```
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
41 42 43 01 02 FF 09 0B 08 7D 28 29 2D 3D 64 65
66 0A
```

Windows下运行，文件大小：___19_____
输出的c是：__18_____
Linux下运行，文件大小：_____18____
输出的c是：____18___
为什么？ .get() 内部读\xFF 是当 255，并不是EOF（-1），不会与之冲突而误判，Win文件大1是因为 0D0A

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABC\x1\x2\xff\t\v\b\175()-=def"<<endl;
    out.close();

    ifstream in("out.txt", ios::in); //可加ios::binary
    int c=0;
    char ch;
    while((ch=in.get())!=EOF) {
        c++;
        }
    cout << c << endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：___19_____
输出的c是：__5_____
Linux下运行，文件大小：_____18____
输出的c是：____5_____
为什么？.get() 返回是 255，赋值给 char ch 就成了 0xff(char的-1)，再隐式转换成 int (-1)，等于 EOF，就停了

综合例12~例14，结论：当文件中含字符_\x1A_时，不能用十进制方式读取，而当文件中含字符_\xFF_时，是可以用二/十进制方式正确读取的

§17. 输入输出流

本页需填写答案

例15：比较格式化读和read()读的区别，并观察gcount()/tellg()在不同读入方式时值的差别

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ" << endl;
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[30];
    in >> name;
    cout << '*' << name << '*' << endl;
    cout << int(name[26]) << endl;
    cout << in.gcount() << endl;
    cout << in.tellg() << endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：_____28____

输出的name是：_ABCDEFGHIJKLMNOPQRSTUVWXYZ_

name[26]的值是：___0_____

gcount()的值是：___0_____

tellg()的值是：____26_____

说明：in >> 方式读入字符串时，和cin方式相同，都是

读到__空白符__停止，并在数组最后加入一个__\0__。

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ" << endl;
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[30];
    in.read(name, 26);
    cout << '*' << name << '*' << endl;
    cout << int(name[26]) << endl;
    cout << in.gcount() << endl;
    cout << in.tellg() <<endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：____28_____

输出的name是：ABCDEFGHIJKLMNOPQRSTUVWXYZ???

name[26]的值是：___??_____

gcount()的值是：___26_____

tellg()的值是：____26_____

说明：in.read()读入时，是读到_指定数或者文件结尾_停止，

不在数组最后加入一个__\0__。

综合左右：gcount()仅对_read()_方式读时有效，可返回最后读取的字节数；tellg()则对两种读入方式均__有效___。

§17. 输入输出流

本页需填写答案

例16：比较read()读超/不超过文件长度时的区别，并观察gcount()/tellg()/good()的返回值

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[30] = "000000000000000000000000000000";
    in.read(name, 20);
    cout << '*' << name << '*' << endl;
    cout << int(name[20]) << endl;
    cout << in.gcount() << endl;
    cout << in.tellg() << endl;
    cout << in.good() << endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：____26_____
输出的name是：ABCDEFGHIJKLMNOPQRST00000000
name[20]的值是：___48_____
gcount()的值是：___20_____
tellg()的值是：____20_____
good()的值是：_____1_____

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[30] = "000000000000000000000000000000";
    in.read(name, 200);
    cout << '*' << name << '*' << endl;

    cout << in.gcount() << endl;
    cout << in.tellg() <<endl;
    cout << in.good() << endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：____26_____
输出的name是：ABCDEFGHIJKLMNOPQRSTUVWXYZ000

gcount()的值是：__26_____
tellg()的值是：___-1_____
good()的值是：____0_____

§17. 输入输出流

本页需填写答案

例17：使用seekg()移动文件指针，观察gcount()/tellg()/seekg()在不同情况下的返回值

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[80];
    in.read(name, 10);
    cout << in.tellg() << " " << in.gcount() << endl;
    name[10] = '\0';
    cout << '*' << name << '*' << endl;
    in.seekg(-5, ios::cur);
    cout << in.tellg() << endl;
    in.read(name, 10);
    cout << in.tellg() << " " << in.gcount() << endl;
    name[10] = '\0';
    cout << '*' << name << '*' << endl;
    in.close();
    return 0;
}
```

Windows下运行，输出依次是：

```
10 10
*ABCDEFGHIJ*
5
15 10
*FGHIJKLMNO*
```

```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[80];
    in.read(name, 30);
    cout << in.tellg() << " " << in.gcount() << endl;
    name[30] = '\0';
    cout << '*' << name << '*' << endl;
    in.seekg(5, ios::beg);
    cout << in.tellg() << endl;
    in.read(name, 30);
    cout << in.tellg() << " " << in.gcount() << endl;
    name[30] = '\0';
    cout << '*' << name << '*' << endl;
    in.close();
    return 0;
}
```

Windows下运行，输出依次是：

```
-1 26
*ABCDEFGHIJKLMNOPQRSTUVWXYZ?????*
-1
-1 0
*ABCDEFGHIJKLMNOPQRSTUVWXYZ?????*
```

综合左右：tellg()/gcount()/seekg()仅在 .good()状态正常 情况下返回正确值，因此，每次操作完成后，最好判断流对象自身状态，正确才可继续下一步。

§17. 输入输出流

本页需填写答案

例18：使用seekg()/gcount()/tellg()/good()后判断流对象状态是否正确，若不正确则恢复正确状态后再继续使用

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[80];
    in.read(name, 30);
    cout << in.tellg() << " " << in.gcount() << endl;
    name[30] = '\0';
    cout << '*' << name << '*' << endl;
    if (!in.good())
        in.clear();

    in.seekg(5, ios::beg);
    cout << in.tellg() << endl;
    in.read(name, 30);
    cout << in.tellg() << " " << in.gcount() << endl;
    name[30] = '\0';
    cout << '*' << name << '*' << endl;
    if (!in.good())
        in.clear();
    in.close();
    return 0;
}
```

Windows下运行，输出依次是：

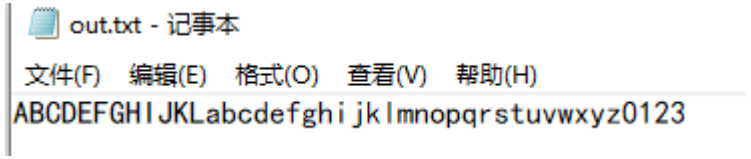
```
-1 26
*ABCDEFGHIJKLMNOPQRSTUVWXYZ???*
5
-1 21
*FGHIJKLMNOPQRSTUVWXYZVWXYZ???*
```

§17. 输入输出流

本页需填写答案

例19：读写方式打开时的seekg()/seekg()同步移动问题

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    fstream file("out.txt", ios::in|ios::out|ios::binary);
    char name[80];
    file.read(name, 30);
    cout << file.tellg() << " " << file.gcount()
                         << " " << file.tellp() << endl;
    name[30] = '\0';
    cout << '*' << name << '*' << endl;
    if (!file.good())
        file.clear();

    file.seekg(5, ios::beg);
    cout << file.tellg() << " " << file.tellp() << endl;

    file.seekp(12, ios::beg);
    cout << file.tellg() << " " << file.tellp() << endl;

    strcpy(name, "abcdefghijklmnopqrstuvwxyz0123");
    file.write(name, 30);
    cout << file.tellg() << " " << file.tellp() << endl;
    file.close();

    return 0;
}
```



Windows下运行，输出依次是：
```
-1 26 -1
*ABCDEFGHIJKLMNOPQRSTUVWXYZ???*
5 5
12 12
42 42
```

结论：

1、读写方式打开时，tellg()/tellp()均可以使用，且读写后两个函数的返回值均相同

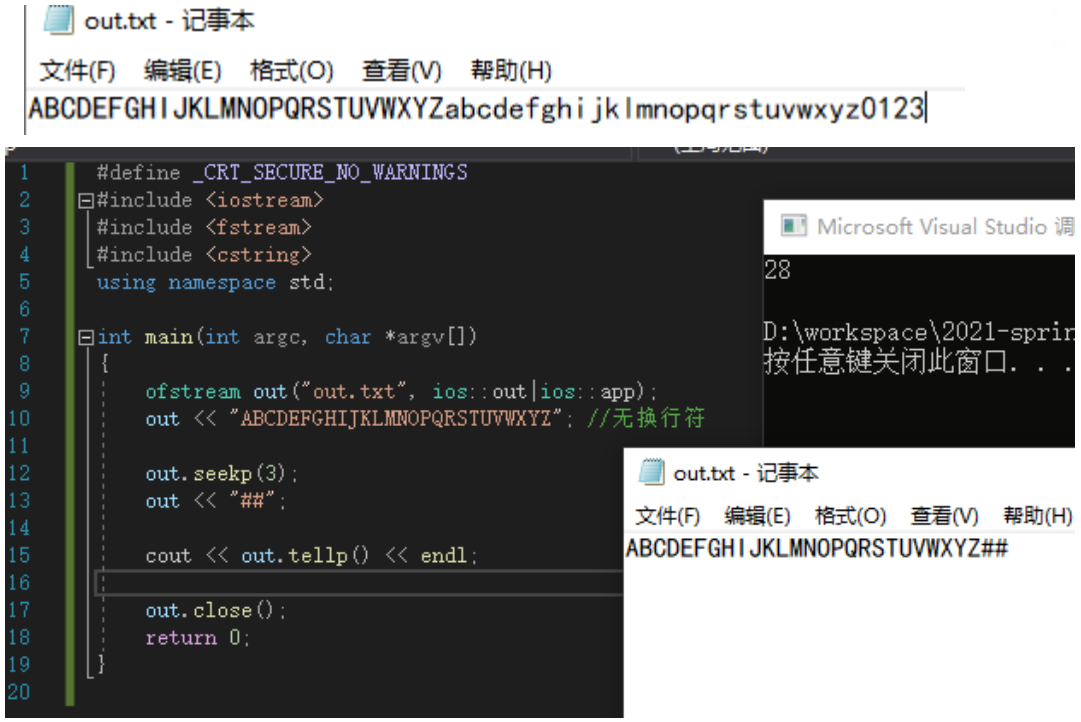
2、文件指针的移动，seekg()/seekp()均可

§17. 输入输出流

本页需填写答案

例20：读写方式打开时加ios::app方式后，读写指针移动及写入问题

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    fstream file("out.txt", ios::in|ios::out|ios::binary|ios::app);
    char name[80];
    file.read(name, 30);
    cout << file.tellg() << " " << file.gcount()
                         << " " << file.tellp() << endl;
    name[30] = '\0';
    cout << '*' << name << '*' << endl;
    if (!file.good())
        file.clear();

    file.seekg(5, ios::beg);
    cout << file.tellg() << " " << file.tellp() << endl;

    file.seekp(12, ios::beg);
    cout << file.tellg() << " " << file.tellp() << endl;

    strcpy(name, "abcdefghijklmnopqrstuvwxyz0123");
    file.write(name, 30);
    cout << file.tellg() << " " << file.tellp() << endl;
    file.close();
    return 0;
}
```



Windows下运行，输出依次是：-1 26 -1
ABCDEFGHIJKLMNOPQRSTUVWXYZ???
5 5
12 12
56 56

结论：

1、加ios::app后，虽然seekg()/seekp()可以移动文件指针，但是写入的位置__是原文件的结尾_____

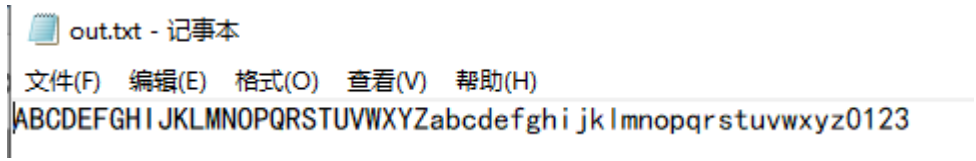
2、自行测试ofstream方式打开加ios::app的情况，与本例的结论____一致_____(一致/不一致)

§17. 输入输出流

本页需填写答案

例20：读写方式打开时加ios::app方式后，读写指针移动及写入问题

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    fstream file("out.txt", ios::in|ios::out|ios::binary|ios::app);
    char name[80];
    file.read(name, 30);
    cout << file.tellg() << " " << file.gcount()
                         << " " << file.tellp() << endl;
    name[30] = '\0';
    cout << '*' << name << '*' << endl;
    if (!file.good())
        file.clear();

    file.seekg(5, ios::beg);
    cout << file.tellg() << " " << file.tellp() << endl;

    strcpy(name, "abcdefghijklmnopqrstuvwxyz0123");
    file.write(name, 30);
    cout << file.tellg() << " " << file.tellp() << endl;
    file.close();

    return 0;
}
```



Windows下运行，输出依次是：-1 26 -1
ABCDEFGHIJKLMNOPQRSTUVWXYZ???
5 5
56 56

结论：加ios::app后，读写方式打开时，tellg()/tellp()均可以使用，且无论读写，两个函数的返回值均相同，表示两个文件指针是同步移动的