§8.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的差异，掌握与文件有关的流函数的正确用法

3、题目明确指定编译器外，缺省使用VS2022即可

★ 如果要换成其他编译器，可能需要自行修改头文件适配

★ 部分代码编译时有warning，不影响概念理解，可以忽略

3、直接在本文件上作答，写出答案/截图（不允许手写、手写拍照截图）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、颜色、文本框的位置等

★ 贴图要有效部分即可，不需要全部内容

★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可

★ 不允许手写在纸上，再拍照贴图

★ 允许在各种软件工具上完成（不含手写），再截图贴图

★ 如果某题要求VS+Dev的，则如果两个编译器运行结果一致，贴VS的一张图即可，如果不一致，则两个图都要贴

4、转换为pdf后提交

5、12月8日前网上提交本次作业（在“文档作业”中提交）

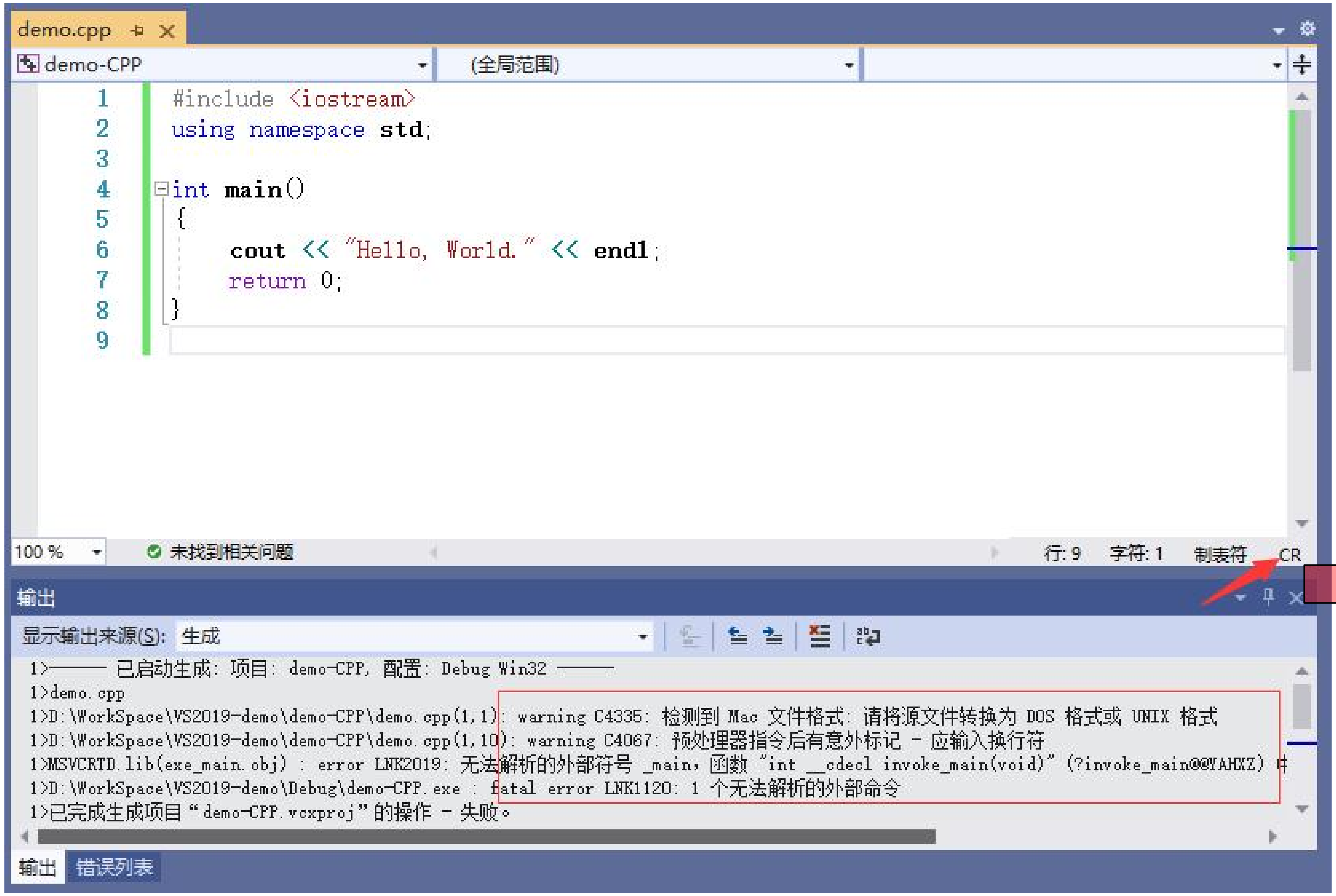
特别说明：

★ 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

§8.输入输出流

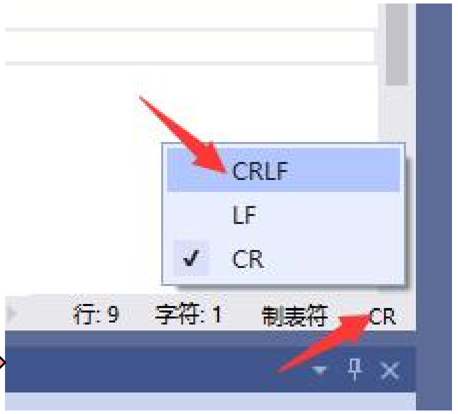
注意：

附1：用WPS等其他第三方软件打开PPT，将代码复制到VS2022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



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

int main()
{
    cout << "Hello, World." << endl;
    return 0;
}

```

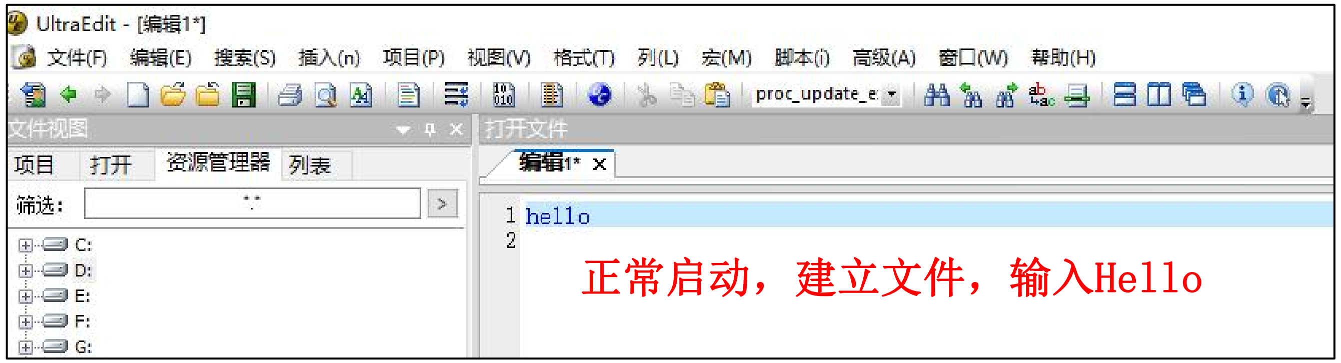
```
1>------ 已启动生成: 项目: demo-CPP, 配置: Debug Win32 ------
1>demo.cpp
1>D:\WorkSpace\VS2019-demo\demo-CPP\demo.cpp(1,1): warning C4335: 检测到 Mac 文件格式: 请将源文件转换为 DOS 格式或 UNIX 格式
1>D:\WorkSpace\VS2019-demo\demo-CPP\demo.cpp(1,10): warning C4067: 预处理器指令后有意外标记 - 应输入换行符
1>MSVCRTD.lib(exe_main.obj) : error LNK2019: 无法解析的外部符号 _main，函数 "int __cdecl invoke_main(void)" (?invoke_main@@YAHXZ) 中
1>D:\WorkSpace\VS2019-demo\Debug\demo-CPP.exe : fatal error LNK1120: 1 个无法解析的外部命令
1>已完成生成项目"demo-CPP.vcxproj"的操作 - 失败。
```



§8. 输入输出流

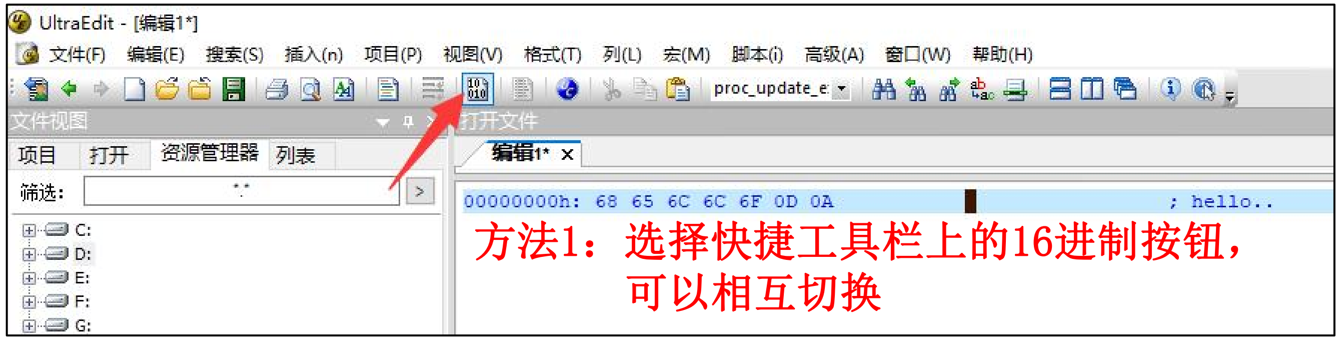
注意：

附2：附件给出的UltraEdit查看文件的16进制形式的方法（三种）


UltraEdit - [编辑1*]
文件(F) 编辑(E) 搜索(S) 插入(n) 项目(P) 视图(V) 格式(T) 列(L) 宏(M) 脚本(i) 高级(A) 窗口(W) 帮助(H)
文件视图
打开文件
项目 打开 资源管理器 列表
筛选:
编辑1*
C: D: E: F: G:

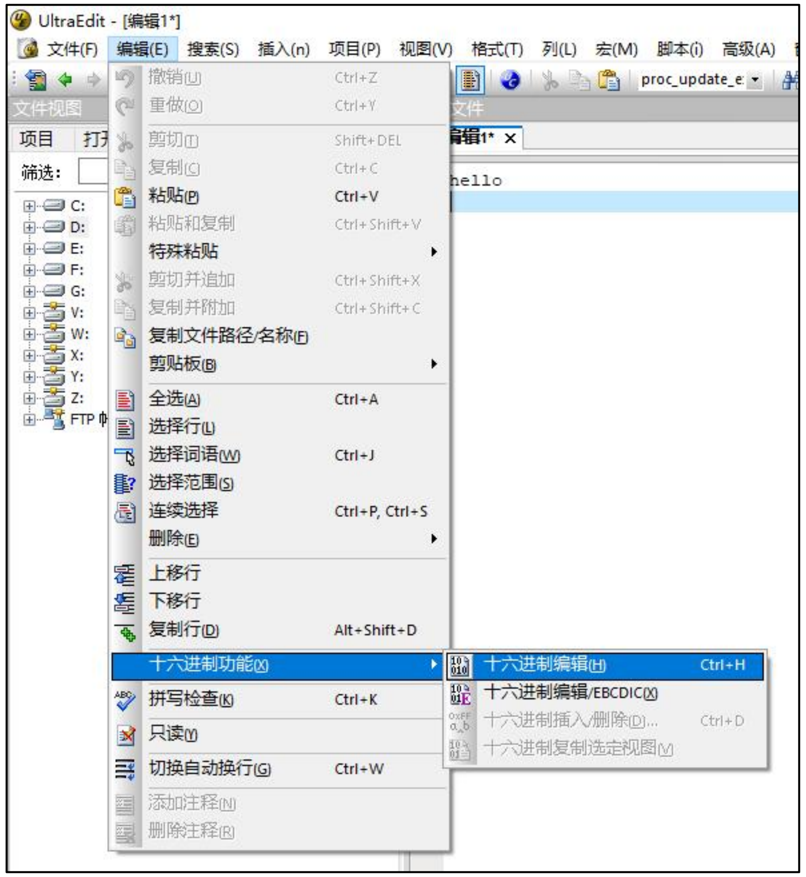
```
hello
```

正常启动，建立文件，输入Hello


UltraEdit - [编辑1*]
文件(F) 编辑(E) 搜索(S) 插入(n) 项目(P) 视图(V) 格式(T) 列(L) 宏(M) 脚本(i) 高级(A) 窗口(W) 帮助(H)
文件视图
打开文件
项目 打开 资源管理器 列表
筛选:
编辑1*
C: D: E: F: G:

```
00000000h: 68 65 6C 6C 6F 0D 0A                    ; hello..
```

方法1：选择快捷工具栏上的16进制按钮，可以相互切换



方法2："编辑"－"十六进制功能"菜单，可以相互切换

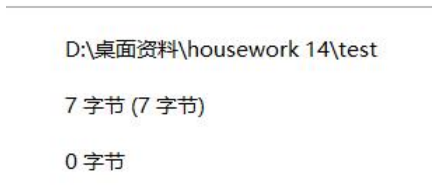
方法3：Ctrl + H 快捷键可以相互切换

§8.输入输出流

本页需填写答案

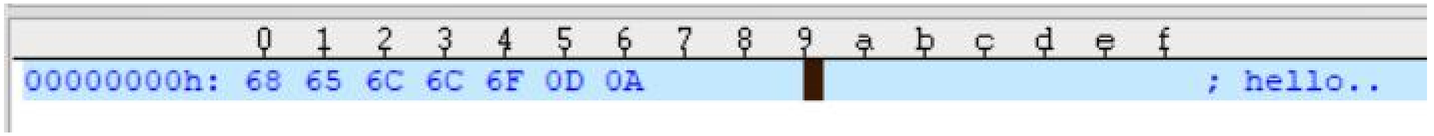
例1：十进制方式写

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

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

    out << "hello" << endl;  //去掉endl后再次运行

    out.close();

    return 0;
}
```



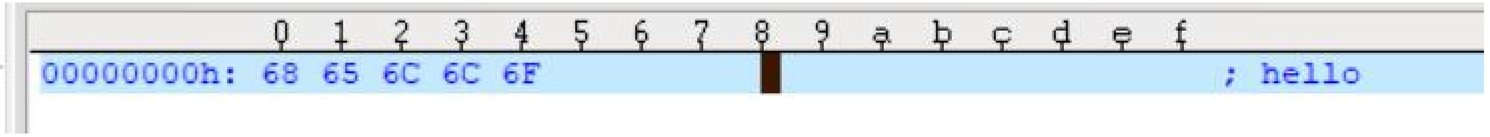
```
D:\桌面资料\housework 14\test
7 字节 (7 字节)
0 字节
```

Windows下运行，out.txt是_7_字节（有endl的情况），用UltraEdit的16进制方式打开的贴图



```
            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，out.txt是_5_字节（无endl的情况），用UltraEdit的16进制方式打开的贴图



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

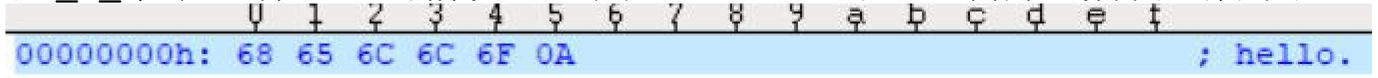
§8. 输入输出流

本页需填写答案

例2：二进制方式写

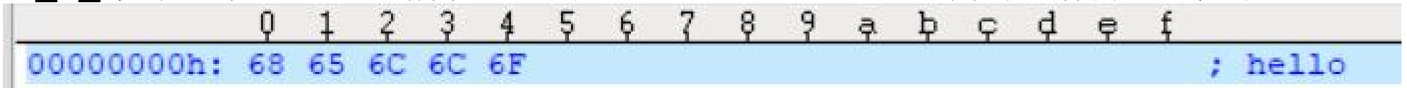
```
#include <iostream>
#include <fstream>
using namespace std;

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

    out << "hello" << endl;  //去掉endl后再次运行

    out.close();

    return 0;
}
```

Windows下运行，out.txt是_6_字节（有endl的情况），用UltraEdit的16进制方式打开的贴图



```
             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，out.txt是_5_字节（无endl的情况），用UltraEdit的16进制方式打开的贴图



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

综合例1/2，endl在十进制和二进制方式下有无区别？
在十进制文件下，endl会转换为\r\n，而在二进制文件下，只会转换为\n;

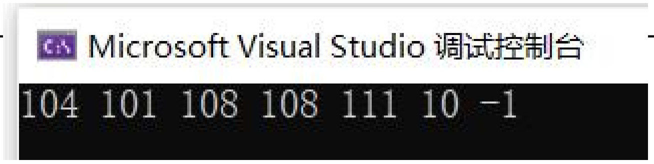
§8. 输入输出流

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：



```
104 101 108 108 111 10 -1
```

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_。

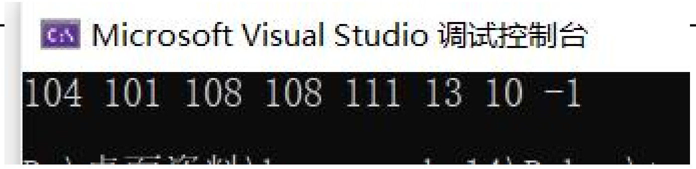
§8.输入输出流

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;
}
```

Windows下运行，输出结果是：


Microsoft Visual Studio 调试控制台

```
104 101 108 108 111 13 10 -1
```

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_。

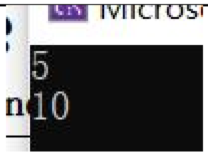
§8.输入输出流

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：

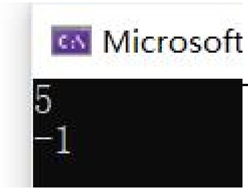


```
5
10
```

说明：in>>str读到_\r_就结束了，_\r\n_还被留在缓冲区中，因此in.peek()读到了_10_。

```
#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：



```
5
-1
```

说明：in.getline读到_\r_就结束了，_\r\n_被读掉，因此in.peek()读到了_EOF_。

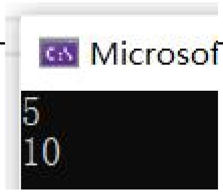
§8.输入输出流

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;
}
```

Windows下运行，输出结果是：

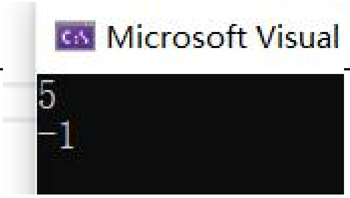

Microsof

```
5
10
```

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 | 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：


Microsoft Visual

```
5
-1
```

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

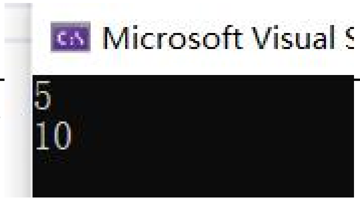
§8.输入输出流

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;
}
```

Windows下运行，输出结果是：

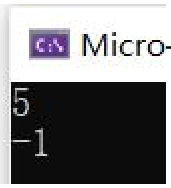


```
5
10
```

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 | 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;
}
```

Windows下运行，输出结果是：



```
5
-1
```

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

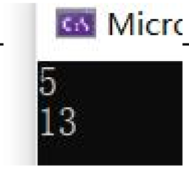
§8.输入输出流

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;
}
```

Windows下运行，输出结果是：

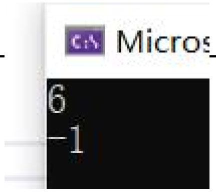

Micrc

```
5
13
```

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

```
#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;
}
```

Windows下运行，输出结果是：


Micros

```
6
-1
```

说明：
1、in.getline读到_\n_就结束了，_\n_被读掉，因此in.peek()读到了_eof_。
2、strlen(str)是_6_，最后一个字符是_\r__

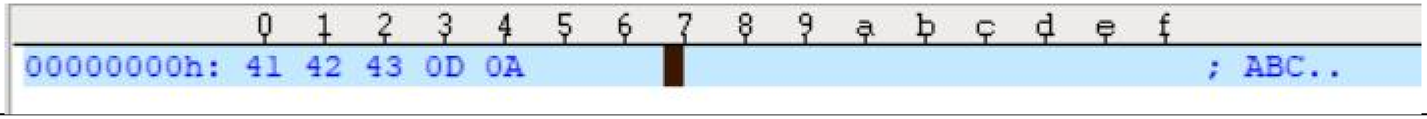
§8.输入输出流

本页需填写答案

例9：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，为什么？
读到尾0就不会继续读了，会将\0转化为\r\n，写入文件



```
            0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 41 42 43 0D 0A                                   ; ABC..
```

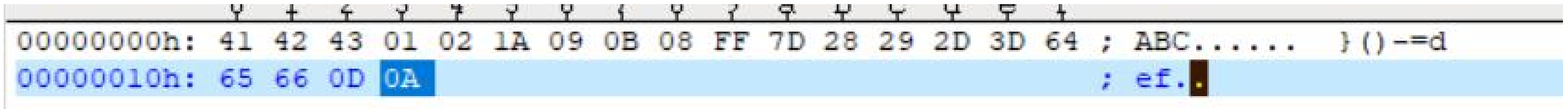
§8.输入输出流

本页需填写答案

例10：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：



```
00000000h: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC......  }()-=d
00000010h: 65 66 0D 0A                                     ; ef..
```

§8.输入输出流

本页需填写答案

例11：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

```cpp
#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;
}
```

Windows下运行，文件大小：__20字节___
　　　　　　　　输出的c是：__6__

为什么？
\x1A 的asall码值是26 对应ctrl+Z 读到了文件终止符退出循环，所以c=6;

```cpp
#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字节___
　　　　　　　　输出的c是：_21_

c的大小比文件大小大__1_，原因是：_
刚开始in.eof()函数是0，读到eof时才为1，但是需要下一轮循环判断，所以大一_

§8.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;
}
```

Windows下运行，文件大小：_19字节_
输出的c是：_5_

为什么？
在这里读到eof就将跳出循环了，并不会让C+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\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字节__
输出的c是：_5__

为什么？
这里读到eof就将跳出循环了，并不会让C+1

§8.输入输出流

本页需填写答案

例13：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;
}
```

Windows下运行，文件大小：_19字节_
　　　　　　　输出的c是：__18__

为什么？
采用in.get()并没有将\ff看做eof，继续读取了，直到读到最后的eof

```
#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字节_
　　　　　　　输出的c是：__5__

为什么？
采用ch=in.get()将\ff看做eof，没有继续读取

综合例11~例13，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

§8.输入输出流

本页需填写答案

例14：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字节__
　　　　　　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__最后一个字符_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字节_
　输出的name是：_ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫烫烫烫烫烫HNe
　　　　　　name[26]的值是：__-52_
　　　　　　gcount()的值是：_26__
　　　　　　tellg()的值是：_26_
说明：in.read()读入时，是读到_最后一个字符_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_。

§8.输入输出流

本页需填写答案

例15：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] = "00000000000000000000000000000";
    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字节__
输出的name是：_ABCDEFGHIJKLMNOPQRST000000000_
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] = "00000000000000000000000000000";
    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字节__
输出的name是：
_ABCDEFGHIJKLMNOPQRSTUVWXYZ000__
gcount()的值是：_26__
tellg()的值是：__-1___
good()的值是：___0___

§8.输入输出流

本页需填写答案

例16：使用seekg()移动文件指针，观察gcount()/tellg()/seekg()在不同情况下的返回

```
#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烫烫_
                          -1_
                          _-1 0_
                          ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫
```

综合左右：tellg()/gcount()/seekg()仅在_读取正确_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。

§8.输入输出流

本页需填写答案

例17：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烫烫_
_5_
_-1 21_
FGHIJKLMNOPQRSTUVWXYZVWXYZ烫烫
```

§8.输入输出流

本页需填写答案

例18：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烫烫_
_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

§8.输入输出流

本页需填写答案

例19：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烫烫_
_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_是按照gcount()的值来写的_

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)

§8.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烫烫_
_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

§8.输入输出流

本页需填写答案

例21：不同打开方式下文件指针的初始值问题

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

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

    cout << "请查看当前out.txt文件的大小" << endl;
    system("pause");

    out.open("out.txt", ios::out | ios::app);
    cout << out.tellp() << endl;
    out << "0123456789";
    cout << out.tellp() << endl;
    out.close();

    return 0;
}
```

Windows下运行，
1、执行到system("pause")的时候，out.txt的大小是：_26字节_
2、加ios::app后，写方式打开，tellp()为_0_，
写入是在文件_结束_(开始/结束)位置，
完成后tellp()是_36_

§8.输入输出流

本页需填写答案

例22：不同打开方式下文件指针的初始值问题

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

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

    cout << "请查看当前out.txt文件的大小" << endl;
    system("pause");

    out.open("out.txt", ios::out | ios::ate);
    cout << out.tellp() << endl;
    out << "0123456789";
    cout << out.tellp() << endl;
    out.close();

    return 0;
}
```

Windows下运行，
1、执行到system("pause")的时候，out.txt的大小是：_26字节__
2、加ios::ate后，写方式打开，tellp()为_0_，
   写入是在文件_开始_(开始/结束)位置，
   完成后tellp()是__10__

注：ate = at end

§8.输入输出流

本页需填写答案

例23：不同打开方式下文件指针的初始值问题

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

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

    cout << "请查看当前out.txt文件的大小" << endl;
    system("pause");

    out.open("out.txt", ios::out | ios::ate | ios::app);
    cout << out.tellp() << endl;
    out << "0123456789";
    cout << out.tellp() << endl;
    out.close();

    return 0;
}
```

Windows下运行，
1、执行到system("pause")的时候，out.txt的大小是：_26字节_
2、同时加ios::ate|ios::app后，写方式打开，tellp()为_26_，
   写入是在文件_结束_(开始/结束)位置，
   完成后tellp()是_36_

结论：结合本例及前两例，ios::ate加在ofstream方式的输出文件上
      _有_(有/无)实用价值

§8.输入输出流

本页需填写答案

例24：不同打开方式下文件指针的初始值问题

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

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

    cout << "请查看当前out.txt文件的大小" << endl;
    system("pause");

    ifstream in("out.txt", ios::in);
    cout << in.tellg() << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

Windows下运行，
1、执行到system("pause")的时候，out.txt的大小是：_26字节__
2、正常读方式打开，tellg()和peek()为_0_和__65_，
   表示从文件的_开始_(开始/结束)位置读

§8.输入输出流

本页需填写答案

例25：不同打开方式下文件指针的初始值问题

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

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

    cout << "请查看当前out.txt文件的大小" << endl;
    system("pause");

    ifstream in("out.txt", ios::in | ios::ate);
    cout << in.tellg() << endl;
    cout << in.peek() << endl;
    in.close();

    return 0;
}
```

Windows下运行，
1、执行到system("pause")的时候，out.txt的大小是：_26字节__
2、加ios::ate后，读方式打开，tellg()和peek()为_26__和_-1_，
   表示从文件的_结束_(开始/结束)位置读

结论：
1、结合本例及上例，ios::ate加在ifstream方式的输出文件上
   __有_(有/无)实用价值
2、为了避免细节记忆错误，另一种做法是，舍弃ios::ate特性不同，
   在需要读写时直接用seekg()/seekp()自行移动文件开头/结尾，
   你是否_赞成_(赞成/反对)这种做法