§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

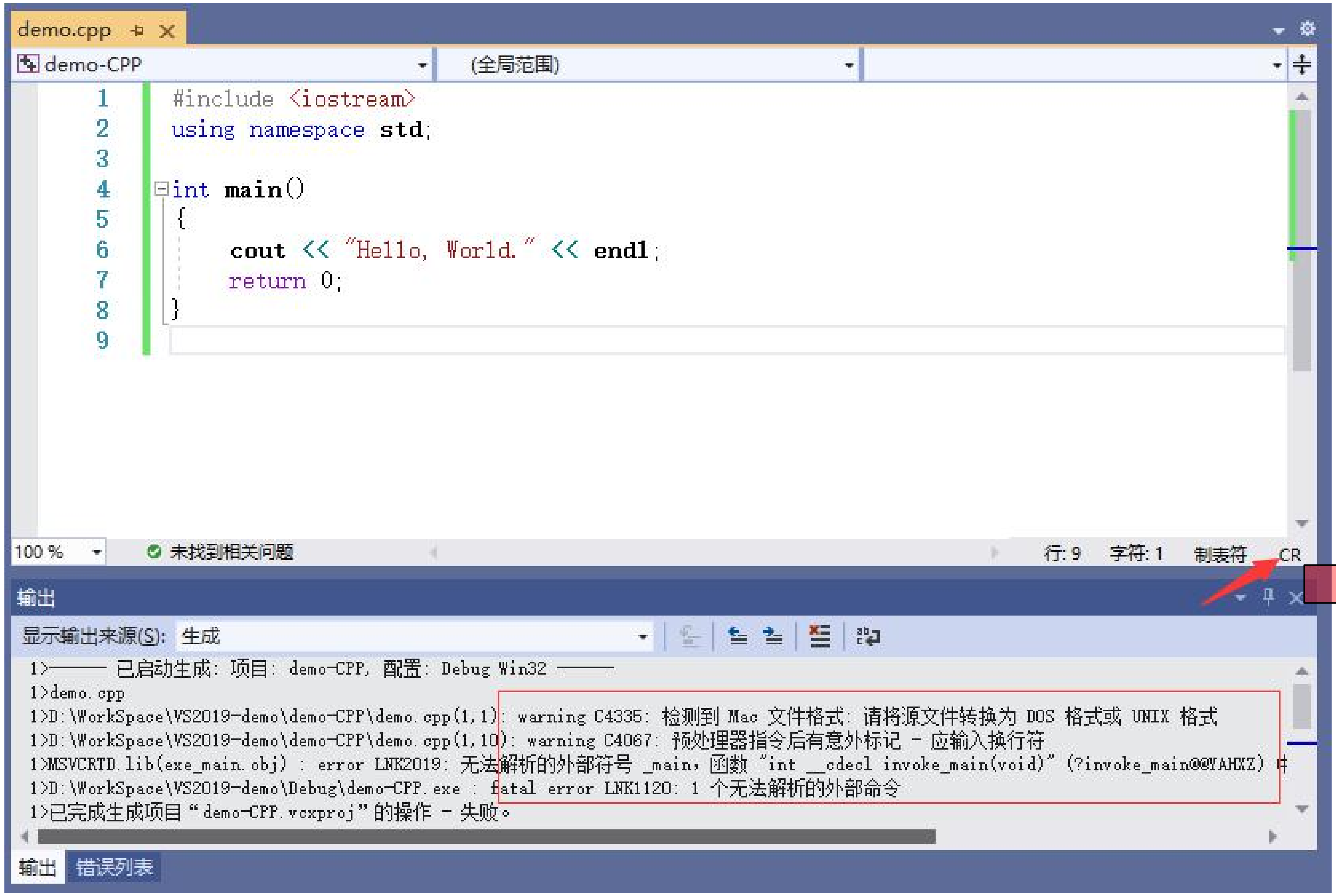
5、无特殊说明，Windows下用VS2019编译

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、6月9日前网上提交本次作业（在“实验报告”中提交）

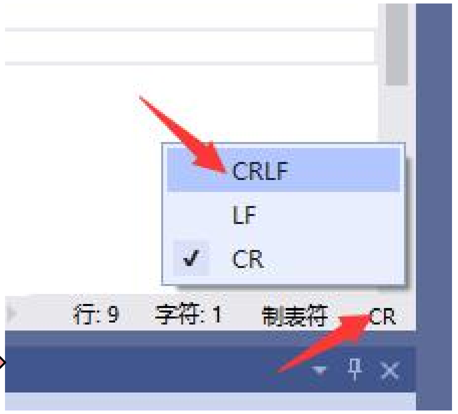
§8. 输入输出流

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



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

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

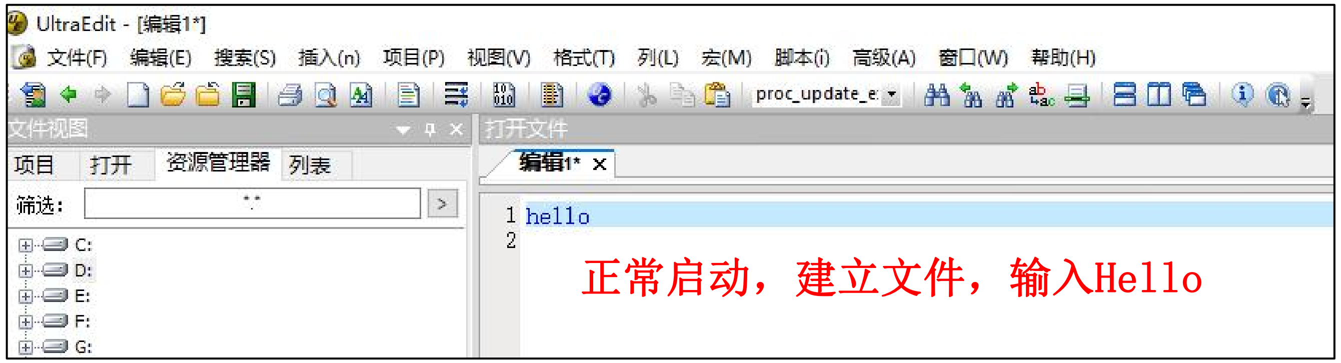
```
1>------ 已启动生成: 项目: demo-CPP, 配置: Debug Win32 ------
1>demo.cpp
1>D:\WorkSpace\VS2019-demo\demo-CPP\demo.cpp(1,1): warning C4335: 检测到 Mac 文件格式: 请将源文件转换为 DOS 格式或 UNIX 格式
1>D:\WorkSpace\VS2019-demo\demo-CPP\demo.cpp(1,10): warning C4067: 预处理器指令后有意外标记 - 应输入换行符
1>MSVCRTD.lib(exe_main.obj) : error LNK2019: 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

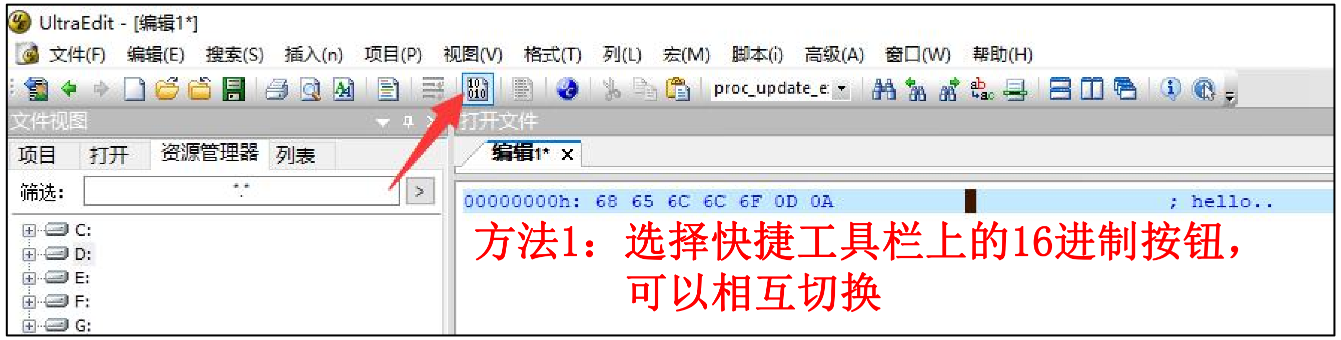
注意：

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)
proc_update_e
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:

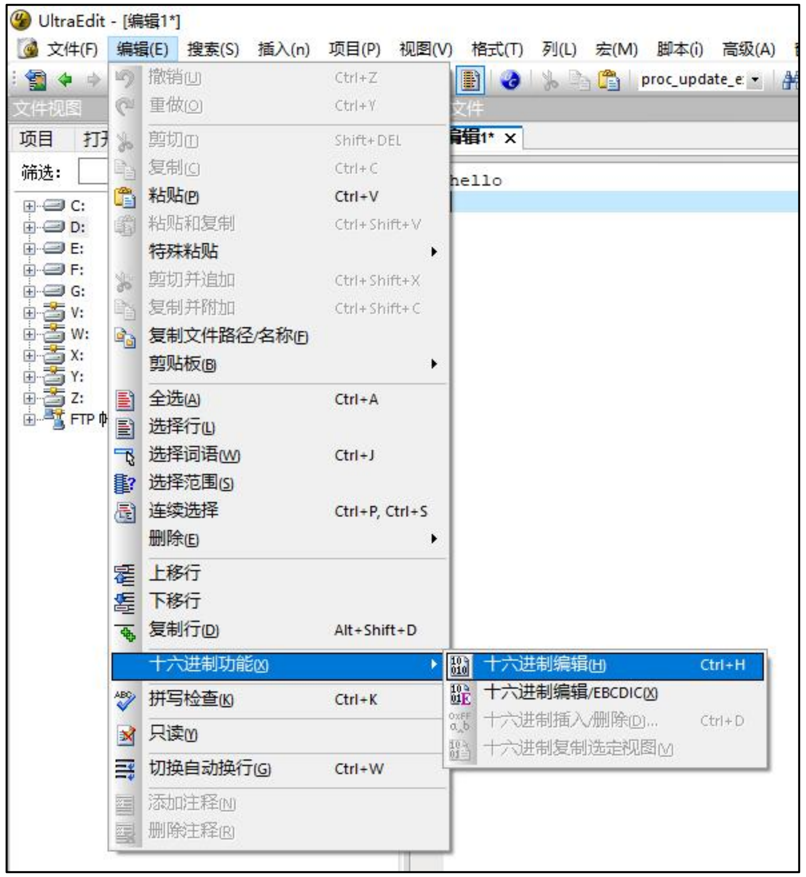
```
1 hello
2
```

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)
proc_update_e
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

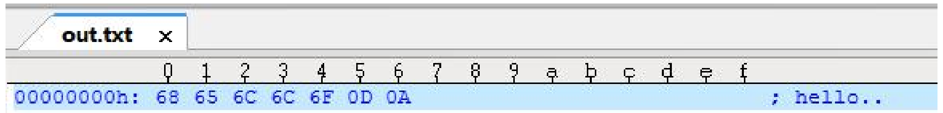
§8.输入输出流

本页需填写答案

例1：十进制方式写

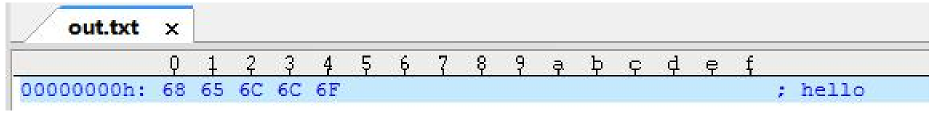
```
#include <iostream>
#include <fstream>
using namespace std;

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;
}
```

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



```
out.txt  ×
           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



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

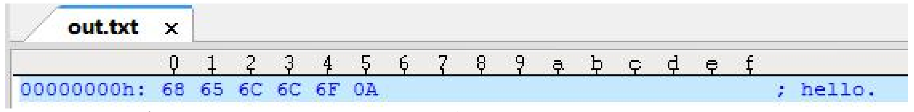
§8.输入输出流

本页需填写答案

例2：二进制方式写

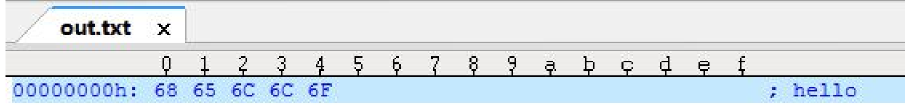
```
#include <iostream>
#include <fstream>
using namespace std;

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



```
out.txt ×
            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



```
out.txt ×
            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？

有区别。十进制时，endl在十六进制中查看为0D 0A,而二进制时查看为0A

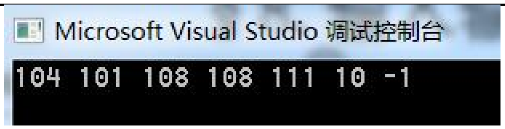
§8.输入输出流

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____。

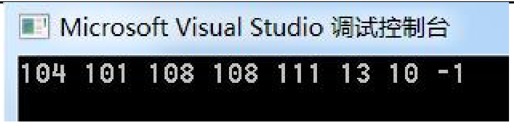
§8.输入输出流

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____。

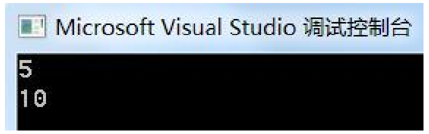
§8.输入输出流

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

```cpp
#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：

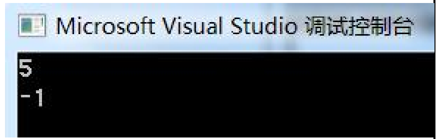


```
5
10
```

说明：in>>str读到__o__就结束了，_换行符___还被留在缓冲区中，因此in.peek()读到了_换行符_____。

```cpp
#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_o__就结束了，_换行符___被读掉，因此in.peek()读到了__结束符___。

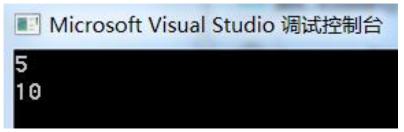
§8.输入输出流

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：

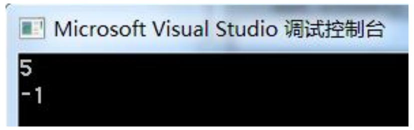


```
5
10
```

说明：in>>str读到__o__就结束了，_换行符___还被留在缓冲区中，因此in.peek()读到了_换行符_____。

```
#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：



```
5
-1
```

说明：in.getline读到__o__就结束了，_换行符___被读掉，因此in.peek()读到了__结束符___。

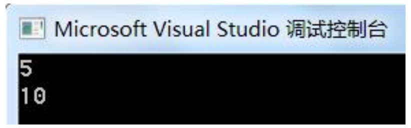
§8.输入输出流

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：

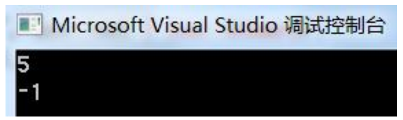

Microsoft Visual Studio 调试控制台

```
5
10
```

说明：in>>str读到__o__就结束了，_换行符___还被留在缓冲区中，因此in.peek()读到了__换行符____。

```
#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：


Microsoft Visual Studio 调试控制台

```
5
-1
```

说明：in.getline读到__o__就结束了，_换行符___被读掉，因此in.peek()读到了__结束符___。

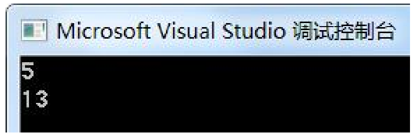
§8.输入输出流

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：

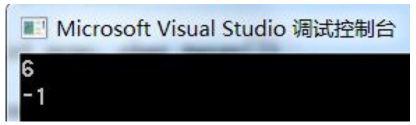


```
5
13
```

说明：in>>str读到__o__就结束了，__回车符__还被留在缓冲区中，因此in.peek()读到了__回车符___。

```
#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：



```
6
-1
```

说明：

1、in.getline读到__回车符__就结束了，__\r__被读掉，因此in.peek()读到了__结束符___。

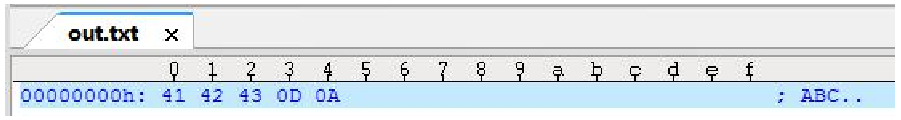
2、strlen(str)是__6___，最后一个字符是_\r__

§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;
}
```



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

Windows下运行，out.txt的大小是__5___字节，为什么？
写入时遇到\0时自动结束了写入，所以只写进\0之前的5个字节。

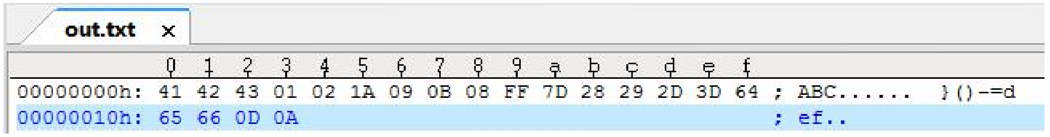
§8. 输入输出流

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：



```
out.txt  ×
              0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
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

```
#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？

前5次正常读取，第6次时读到CTRL+Z，自动结束。故循环只进行了6次。

```
#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：__全部字符读取完后才能读到结束符，此时又多读取了一次。___

§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;
}
```
	```
	#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？ 循环结束条件是正常读到了字符，in.get（）不为-1，前5次正常读取，第6次读到了CTRL+Z，循环结束，故循环共进行了5次，c=5.	Windows下运行，文件大小：____19字节_______ 输出的c是：____5______ 为什么？ 循环结束条件是ch不为-1，前5次正常读取，第6次读到了CTRL+Z，askii码为-1，循环结束，故循环共进行了5次，c=5.

§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？
读到\xff时，正确读取到了“-1”这个值，所以不会停止，而文件最后有结束符，读到时将in.get()的状态置为-1，此时停止，共进行了文件大小减1次循环，c=18.

```
#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？
读到\xff时，ch=-1，再判断ch的值等于EOF（即-1），循环停止，此时共进行5次循环，c=5.

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__换行符____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??__
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___换行符___停止，
不在数组最后加入一个___\0___。

综合左右：gcount()仅对__二进制__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烫烫*__________
___-1_________
___-1_0_________
__*ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫*______

综合左右：tellg()/gcount()/seekg()仅在___流对象自身状态正确________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_______在原有文件尾部____________________

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