



§ 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

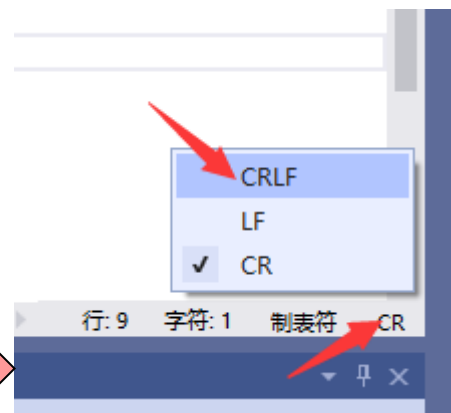
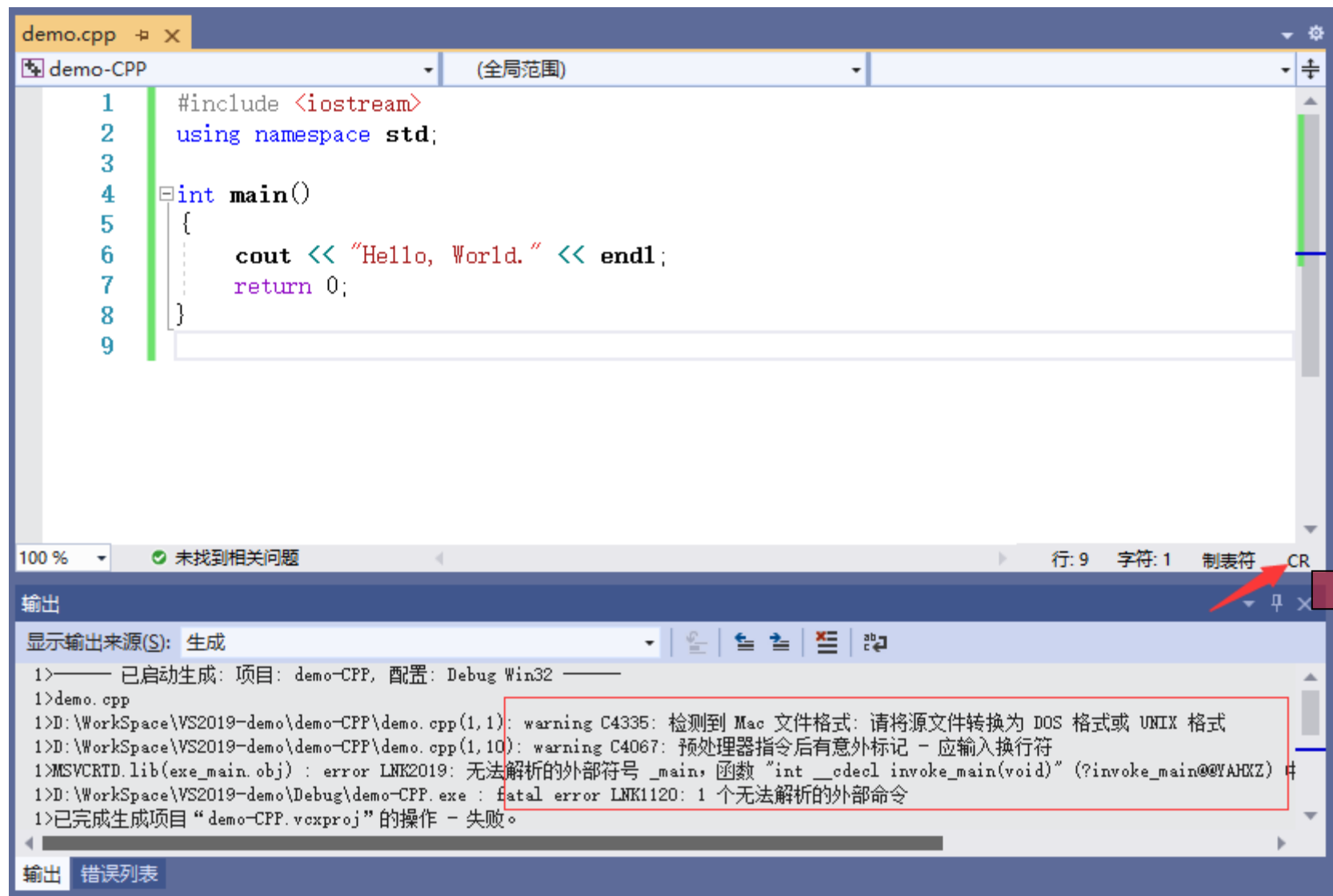


§ 8. 输入输出流

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

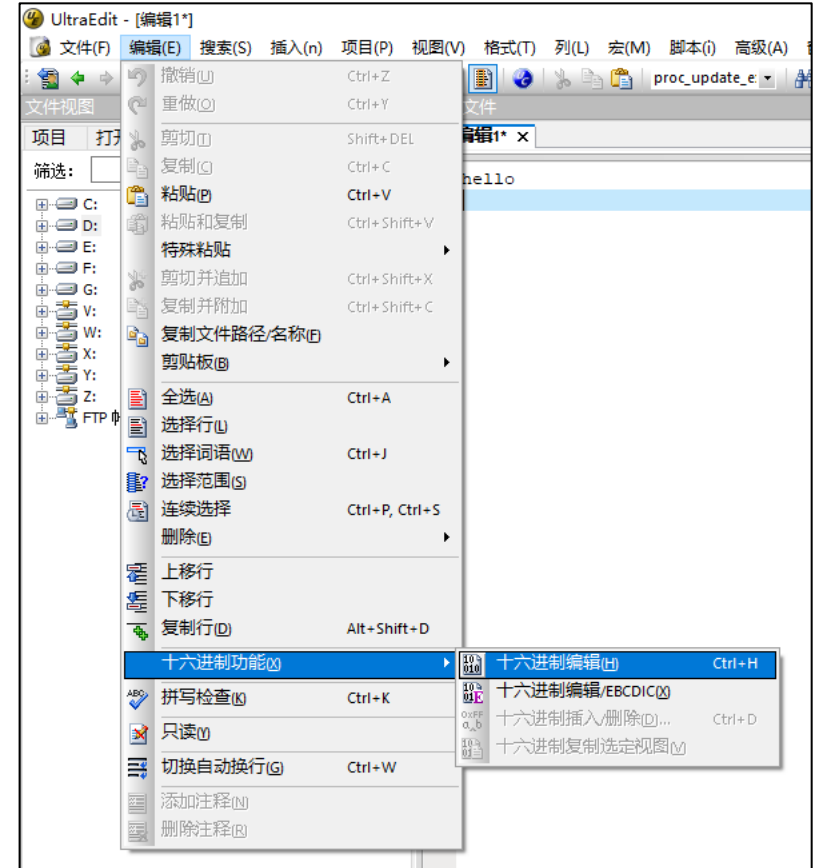
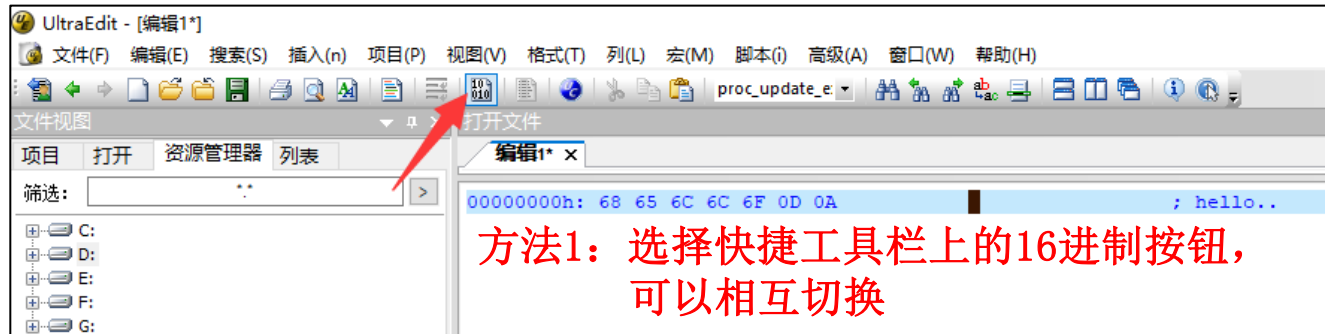
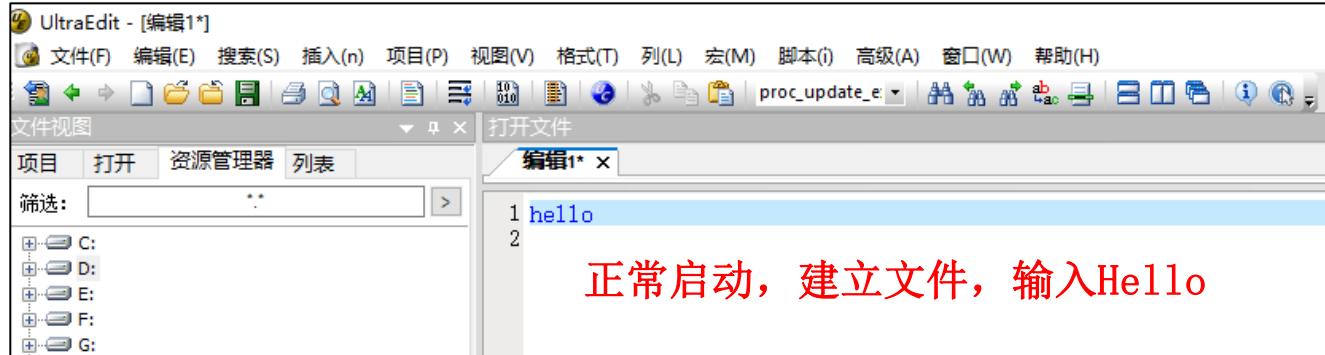




§ 8. 输入输出流

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)



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

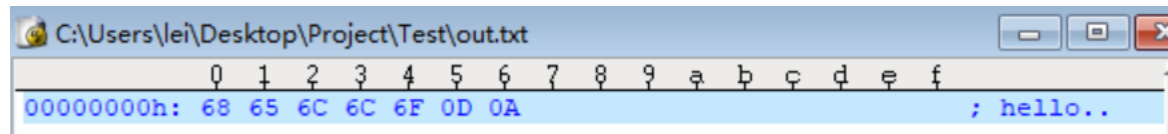
```
#include <iostream>
#include <fstream>
using namespace std;

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

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

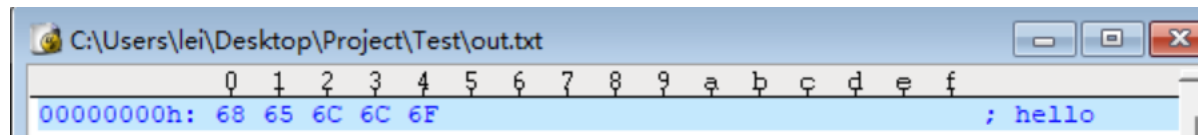
    out.close();

    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

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



§ 8. 输入输出流

本页需填写答案



例2: 二进制方式写

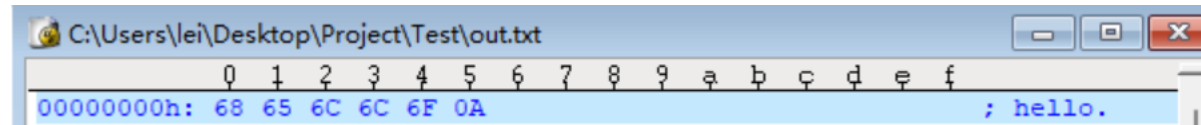
```
#include <iostream>
#include <fstream>
using namespace std;

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

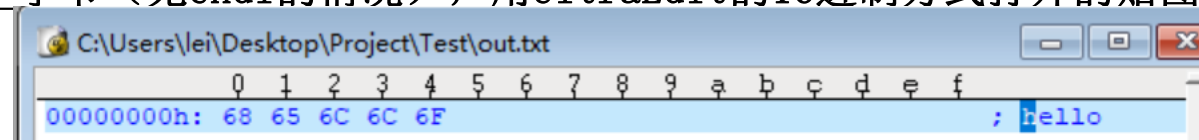
    out.close();

    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

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



综合例1/2, endl在十进制和二进制方式下有无区别？ 有区别，十进制输入有endl的文件有7字节和二进制打开只有6字节

§ 8. 输入输出流

本页需填写答案



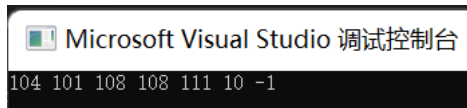
例3：十进制方式写，十进制方式读，0D0A(即“\r\n”)在Windows下的表现

```
#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：



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：

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：

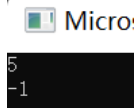
说明：in>>str读到__字母o__就结束了，__\n__还被留在缓冲区中，因此in.peek()读到了__\n__。

```
#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：


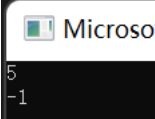
说明：in.getline读到__字母o__就结束了，__\n__被读掉，因此in.peek()读到了__eof__。

§ 8. 输入输出流

本页需填写答案



例6：二进制方式写，十进制方式读，不同读方式在Windows下的表现

<pre>#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; }</pre>	<pre>#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; }</pre>
<p>Windows下运行，输出结果是：</p> 	<p>Windows下运行，输出结果是：</p> 
<p>说明：in>>str读到__字母o__就结束了，__\n__还被留在缓冲区中，因此in.peek()读到了__\n__。</p>	<p>说明：in.getline读到__字母o__就结束了，__\n__被读掉，因此in.peek()读到了__eof__。</p>

§ 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

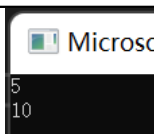
```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

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：



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

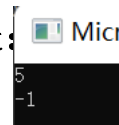
```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

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：



说明：in.getline读到__字母o__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：



说明：in>>str读到__字母o__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：



说明：

- 1、in.getline读到__字母o__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

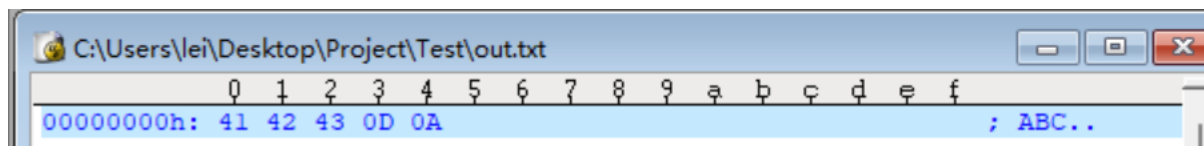


例9：用十进制方式写入含\0的文件，观察文件长度

```
#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？

以十进制方式写入含\0的文件时，遇到\0即停止写入
文件中含0D、0A的原因：cout << “xxx” 返回cout=流对象本身，即cout<<end，写入换行符
会写入两个字节，\r和\n

§ 8. 输入输出流

本页需填写答案



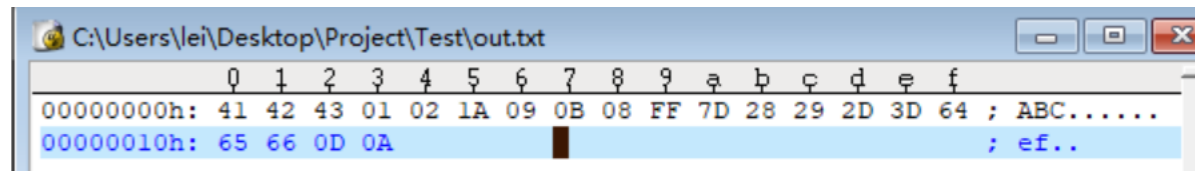
例10：用十进制方式写入含非图形字符(ASCII码32是空格，33-126为图形字符)，但不含\0

```
#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：



§ 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? 十进制读取时, 读到\x1A相当于文件尾, in.eof()返回1, 本次继续, 下次循环终止

```
#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: 22字节
输出的c是: 23

c的大小比文件大小大1, 原因是:

读取最后一个字节后, 还要再读一个字节才跳出循环

§ 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? **ASCII码表中用\x1A表示EOF**
In.get()以十进制方式读到\x1A就表示
遇到了文件尾, 循环终止

```
#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? **ASCII码表中用\x1A表示EOF**
In.get()以十进制方式读到\x1A时, 返回EOF,
Char变量接受EOF, 循环终止

§ 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?

In.get()返回字符的ASCII
读到\xFF是返回255, 循环不终止
读到文件尾是返回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?

In.get()返回字符的ASCII
读到\xFF是返回255, 将其赋值给char变量
Char变量在于EOF比较时, char变量被转成int型, 恰好为-1
循环终止

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

<pre>#include <iostream> #include <fstream> #include <cstring> using namespace std; int main(int argc, char *argv[]) { ofstream out("out.txt", ios::out); out << "ABCDEFGHJKLMNOPQRSTUVWXYZ" << 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; }</pre>	<pre>#include <iostream> #include <fstream> #include <cstring> using namespace std; int main(int argc, char *argv[]) { ofstream out("out.txt", ios::out); out << "ABCDEFGHJKLMNOPQRSTUVWXYZ" << 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; }</pre>
<p>Windows下运行，文件大小： <u>28字节</u></p> <p>输出的name是： <u>ABCDEFGHJKLMNOPQRSTUVWXYZ</u></p> <p>name[26]的值是： <u>\0</u> ASCII为0</p> <p>gcount()的值是： <u>\0</u> ASCII为0</p> <p>tellg()的值是： <u>26</u></p> <p>说明： in >> 方式读入字符串时，和cin方式相同，都是读到 <u>\r</u> 停止，并在数组最后加入一个 <u>\0</u>。</p>	<p>Windows下运行，文件大小： <u>28字节</u></p> <p>输出的name是： <u>ABCDEFGHJKLMNOPQRSTUVWXYZ烫烫烫烫烫烫烫</u></p> <p>name[26]的值是： <u>-52</u></p> <p>gcount()的值是： <u>26</u></p> <p>tellg()的值是： <u>26</u></p> <p>说明： in.read() 读入时，是读到 <u>\r</u> 停止，不在数组最后加入一个 <u>\0</u>。</p>
<p>综合左右： gcount() 仅对 <u>read</u> 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 <u>有效</u>。</p>	

§ 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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[30] = "00000000000000000000000000000000";
    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: ABCDEFGHJKLMNOPQRST000000000
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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    ifstream in("out.txt", ios::in | ios::binary);
    char name[30] = "00000000000000000000000000000000";
    in.read(name, 200);
    cout << '*' << name << '*' << endl;

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


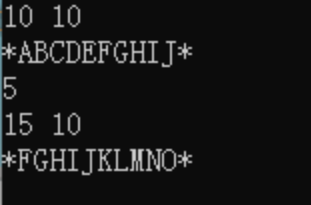

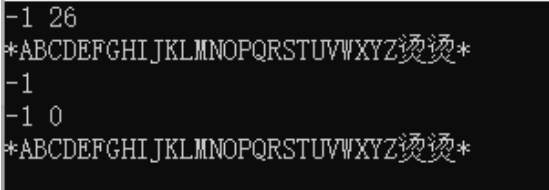
    return 0;
}
```

Windows下运行, 文件大小: 26字节
输出的name是: ABCDEFGHJKLMNOPQRSTUVWXYZ000
gcount()的值是: 26
tellg()的值是: -1
good()的值是: 0



§ 8. 输入输出流

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

<pre>#include <iostream> #include <fstream> #include <cstring> using namespace std; int main(int argc, char *argv[]) { ofstream out("out.txt", ios::out); out << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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; }</pre>	 Microsoft Vis
Windows下运行, 输出依次是:	
<pre>#include <iostream> #include <fstream> #include <cstring> using namespace std; int main(int argc, char *argv[]) { ofstream out("out.txt", ios::out); out << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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; }</pre>	 Microsoft Visual Studio 调试控
Windows下运行, 输出依次是:	
综合左右: tellg()/gcount()/seekg() 仅在__流对象正确时__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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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:

Microsoft Visual Studio 调试器

```
-1 26
*ABCDEFGHJKLMNOPQRSTUVWXYZ烫烫*
5
-1 21
*FGHJKLMNOPQRSTUVWXYZVWXYZ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()/seekp()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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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, "abcdefghijklmnopqrstuvwxy0123");
    file.write(name, 30);
    cout << file.tellg() << " " << file.tellp() << endl;
    file.close();

    return 0;
}
```

Windows下运行, 输出依次是:  Microsoft Visual Studio 调试

```
+1 26 -1
*ABCDEFGHJKLMNOPQRSTUVWXYZ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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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, "abcdefghijklmnopqrstuvwxy0123");
    file.write(name, 30);
    cout << file.tellg() << " " << file.tellp() << endl;
    file.close();
    return 0;
}
```

Windows下运行, 输出依次是:

```
Microsoft Visual Studio 调试
-1 26 -1
*ABCDEFGHJKLMNOPQRSTUVWXYZ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__始终在文件末尾__
- 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)

§ 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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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, "abcdefghijklmnopqrstuvwxy0123");
    file.write(name, 30);
    cout << file.tellg() << " " << file.tellp() << endl;
    file.close();

    return 0;
}
```

Windows下运行, 输出依次是:

Microsoft Visual Studio 调试

```
-1 26 -1
*ABCDEFGHJKLMNOPQRSTUVWXYZ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

(第2问的第2空说结束我觉得也没错, 文件打开后会被清空, 开始位置即结束位置)

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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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
__ 无 __ (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 << "ABCDEFGHJKLMNOPQRSTUVWXYZ"; //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