



§ 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

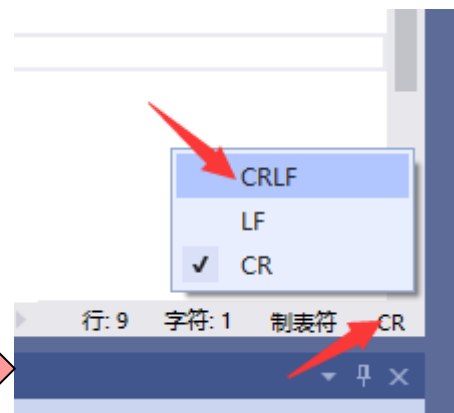
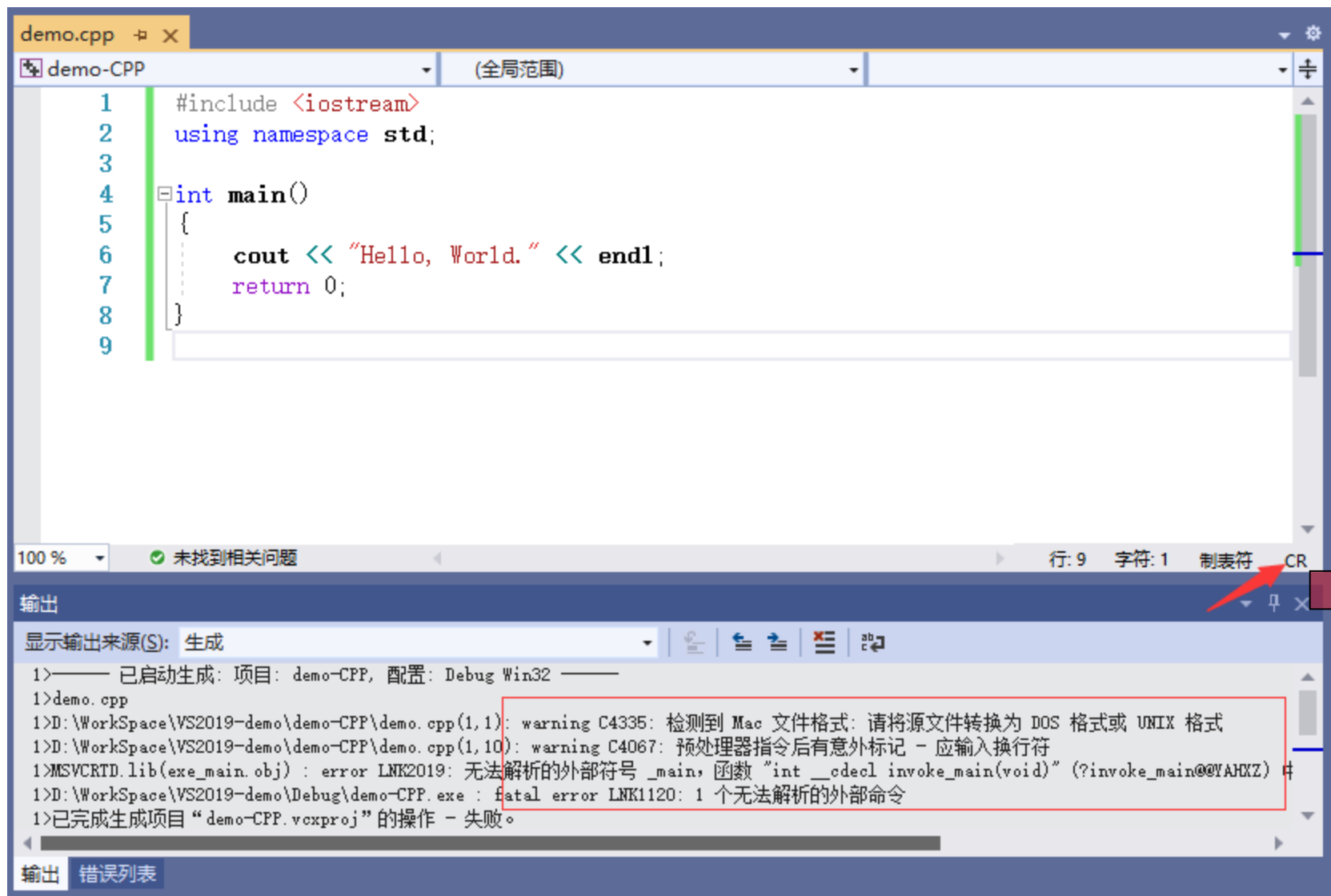


§ 8. 输入输出流

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

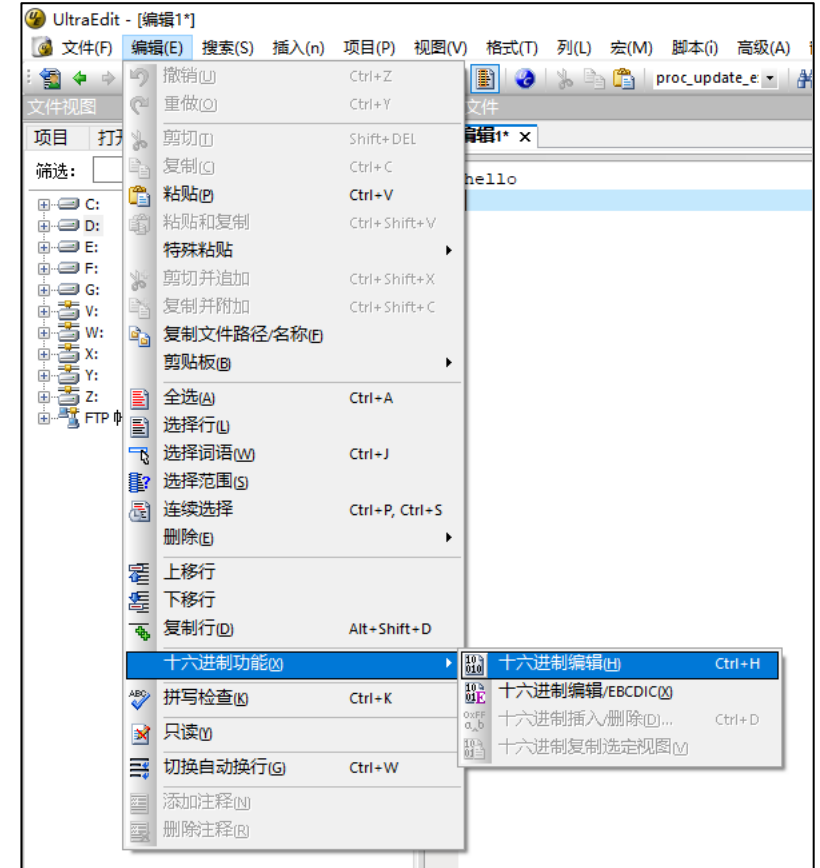
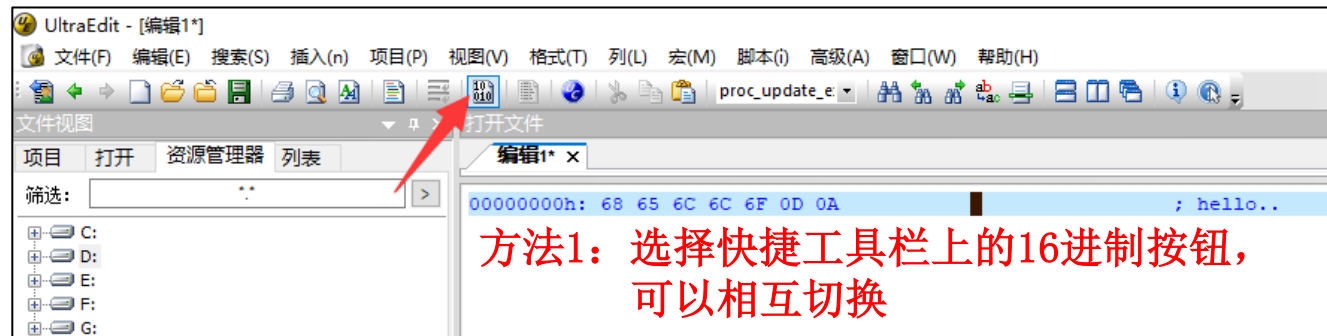
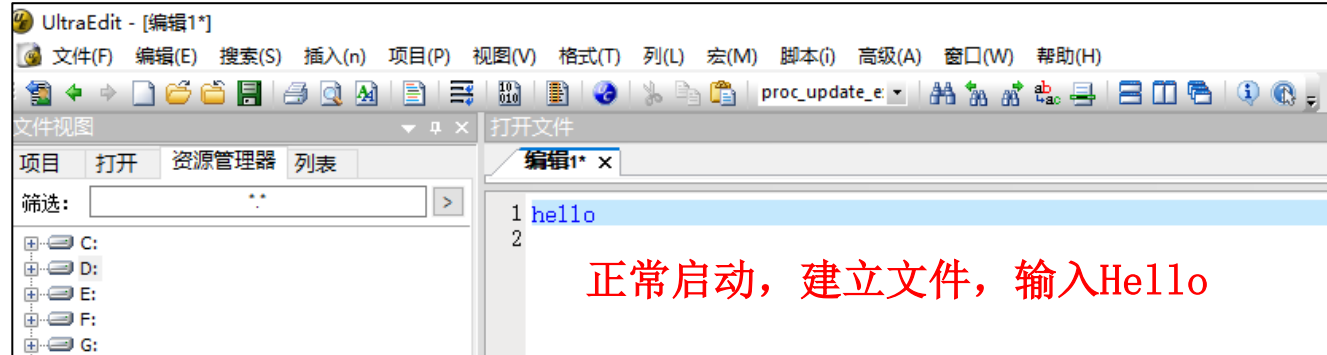




§ 8. 输入输出流

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

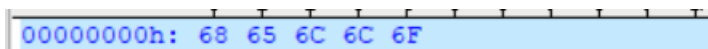
    out.close();

    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?

十进制下endl是2字节0D0A, 而在二进制下面endl是1个字节0A

§ 8. 输入输出流

本页需填写答案

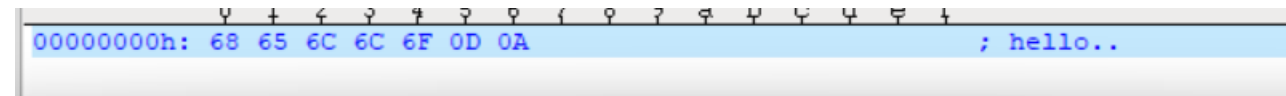


例3：十进制方式写，十进制方式读，0D0A(即“\r\n”)在Windows下的表现

```
#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;
}
```



Microsoft Visual Studio 调试控制台

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:

```
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和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


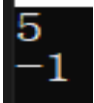
<pre>#include <iostream> #include <fstream> #include <cstring> using namespace std; int main(int argc, char *argv[]) { ofstream out("out.txt", ios::out); out << "hello" << endl; out.close(); char str[80]; ifstream in("out.txt", ios::in); in >> str; cout << strlen(str) << endl; cout << in.peek() << endl; in.close(); return 0; }</pre>	<pre>#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; }</pre>
Windows下运行，输出结果是： <div>5 10</div>	Windows下运行，输出结果是： <div>5 -1</div>
说明：in>>str读到_0D__就结束了，_0A__还被留在缓冲区中，因此in.peek()读到了_0A__。	说明：in.getline读到_0A_就结束了，_0A_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>说明：in>>str读到__o__就结束了，__0A__还被留在缓冲区中，因此in.peek()读到了__0A__。</p>	<p>Windows下运行，输出结果是：</p> <p>说明：in.getline读到__0A__就结束了，__0A__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

<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 ios::binary); 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 ios::binary); in.getline(str, 80); cout << strlen(str) << endl; cout << in.peek() << endl; in.close(); return 0; }</pre>
Windows下运行，输出结果是： <div>5 10</div>	Windows下运行，输出结果是： <div>5 -1</div>
说明：in>>str读到 <u>o</u> 就结束了， <u>0A</u> 还被留在缓冲区中，因此in.peek()读到了 <u>0A</u> 。	说明：in.getline读到 <u>0A</u> 就结束了， <u>0A</u> 被读掉，因此in.peek()读到了 <u>EOF</u> 。

§ 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

<pre>#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; }</pre>	<pre>#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; }</pre>
<p>Windows下运行，输出结果是：5 13</p> <p>说明：in>>str读到_o_就结束了，_0D_还被留在缓冲区中，因此in.peek()读到了_0D_。</p>	<p>Windows下运行，输出结果是：6 -1</p> <p>说明：</p> <p>1、in.getline读到_0A_就结束了，_0A_被读掉，因此in.peek()读到了_EOF_。</p> <p>2、strlen(str)是_6_，最后一个字符是_换行符_。</p>

§ 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？
因为三个字符各占一个字节，0D和0A各占一个字节，
后面三个\x61\x62\x63因为在尾0后不会到文件里面

§ 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:

The screenshot shows the first 20 bytes of the file out.txt in hexadecimal and ASCII. The hex values are: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC..... } () -=d. The ASCII values are: 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

<pre>#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; }</pre>	<pre>#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; }</pre>
<p>Windows下运行，文件大小：<u>20字节</u> 输出的c是：<u>6</u></p> <p>为什么？ 当in.get()读到\x1A时，返回EOF，in.eof()读到EOF，循环结束。</p>	<p>Windows下运行，文件大小：<u>20字节</u> 输出的c是：<u>21</u></p> <p>c的大小比文件大小大<u>1</u>，原因是：<u>0D 0A在二进制下当两个字符处理，当读到\x1A时，一个字节in.eof()依然为假，直到尾部的0D0A时 in.get()读到EOF， in.eof()为真，循环结束。</u></p>

§ 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

<pre>#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; }</pre>	<pre>#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; }</pre>
Windows下运行，文件大小: <u>19字节</u> 输出的c是: <u>5</u>	Windows下运行，文件大小: <u>19字节</u> 输出的c是: <u>5</u>
为什么? 当in.get()读到\0x1A时，返回EOF，	为什么? 两种模式返回x1A的都是一个字节的EOF, 十进制读取也是一个字节的EOF 所以都会在第一个返回

§ 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

<pre>#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; }</pre>	<pre>#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; }</pre>
Windows下运行, 文件大小: <u>19字节</u> 输出的c是: <u>18</u> 为什么? <u>0D 0A在十进制下当成一个字符处理, 当读到0A时, in.get() 返回0A, 所以会再循环, in.get() 读到EOF并返回EOF, 循环结束</u>	Windows下运行, 文件大小: <u>19字节</u> 输出的c是: <u>5</u> 为什么? <u>当in.get() 一个字节的FF时, 返回EOF给ch一个字节, 循环结束。</u>
综合例11~例13, 结论: 当文件中含字符 <u>\x1A</u> 时, 不能用十进制方式读取, 而当文件中含字符 <u>\xFF</u> 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

```
#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;
}
```

Windows下运行, 文件大小: 28字节
输出的name是: ABCDEFGHJKLMNOPQRSTUVWXYZ
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尾零。

```
#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;
}
```

Windows下运行, 文件大小: 28字节
输出的name是: ABCDEFGHJKLMNOPQRSTUVWXYZ烫烫烫烫烫烫0?
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尾零。

综合左右: gcount() 仅对_in.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 << "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: 28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

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

<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>	<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>
Windows下运行，输出依次是： <u>10 10</u> <u>*ABCDEFGHJIJ*</u> <u>5</u> <u>15 10</u> <u>*FGHIJKLMNO*</u>	Windows下运行，输出依次是： <u>-1 26</u> <u>*ABCDEFGHJKLMNOPQRSTUVWXYZ烫烫*</u> <u>-1</u> <u>-1 0</u> <u>*ABCDEFGHJKLMNOPQRSTUVWXYZ烫烫*</u>
综合左右：tellg()/gcount()/seekg() 仅在 <u>read() 不超过文件长度</u> 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:

```
-1 26
*ABCDEFGHJKLMNOPQRSTUVWXYZ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()/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:

-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:

-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__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, "abcdefghijklmnoprstuvwxyz0123");
    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

ABCDEFGHJKLMNOPQRSTUVWXYZ烫烫

5 5

12 12

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 << "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);
    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 << "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 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 << "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);
    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，你是否反对（赞成/反对）这种做法