§15. 输入输出流

要求：

1、安装UltraEdit软件（附件已给，版本虽旧，但够用），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

★ 已安装VSCode的也可通过相关插件进行16进制方式的查看(VSCode某种情况下会自动做格式转换或字符集转换，要注意!!!)

★ 也可以使用其它编辑软件，但不建议NotePad++

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（不强制要求Linux，但建议试一试）

★ 部分代码编译时有warning，不影响概念理解，可以忽略

4、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

5、转换为pdf后提交

6、11月7日前网上提交本次作业（在“文档作业”中提交）

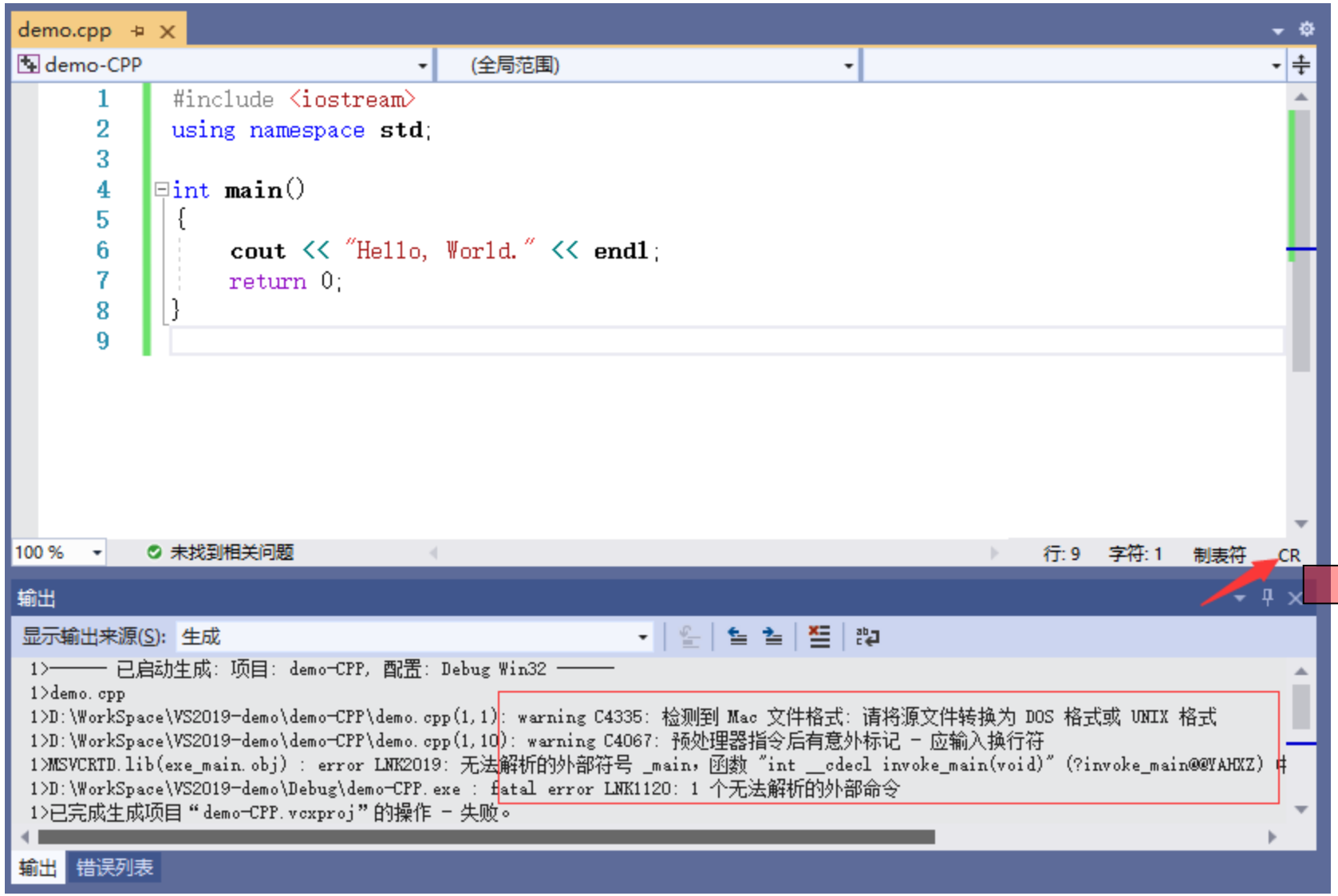
特别说明：

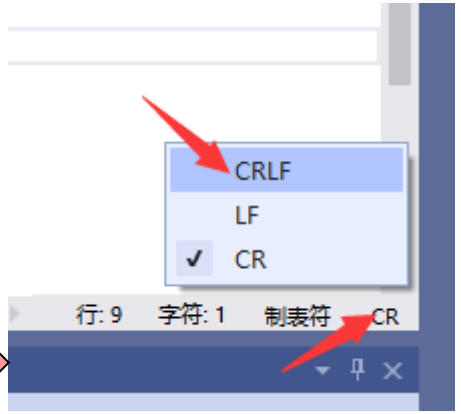
★ 因为篇幅问题，打开文件后均省略了是否打开成功的判断，这在实际应用中是不允许的

§ 15. 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

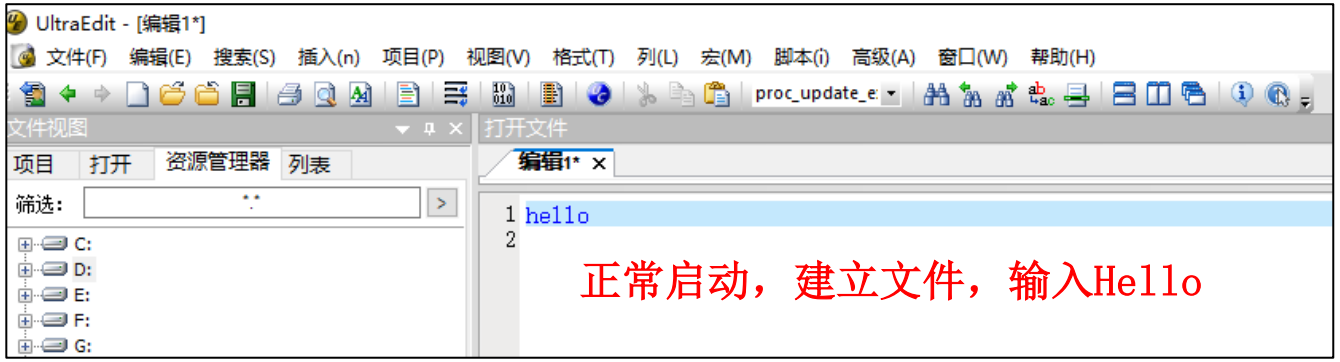




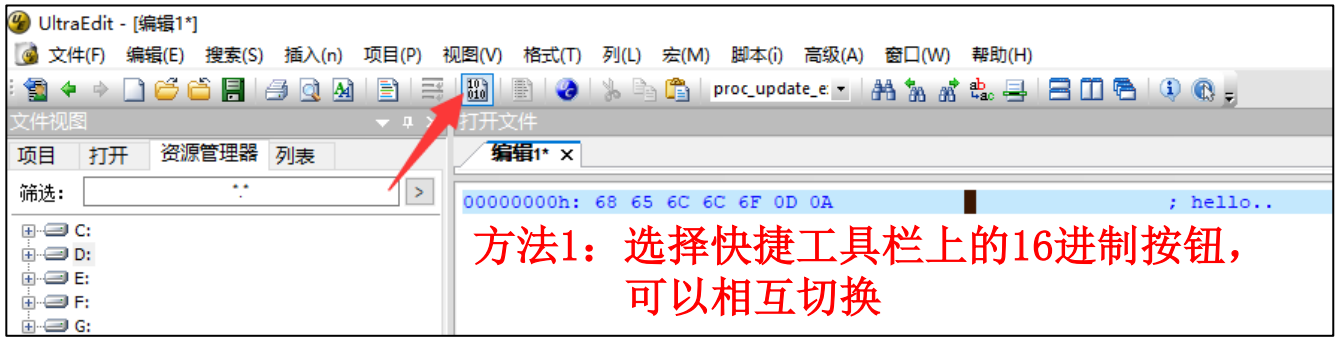
§15. 输入输出流

注意：

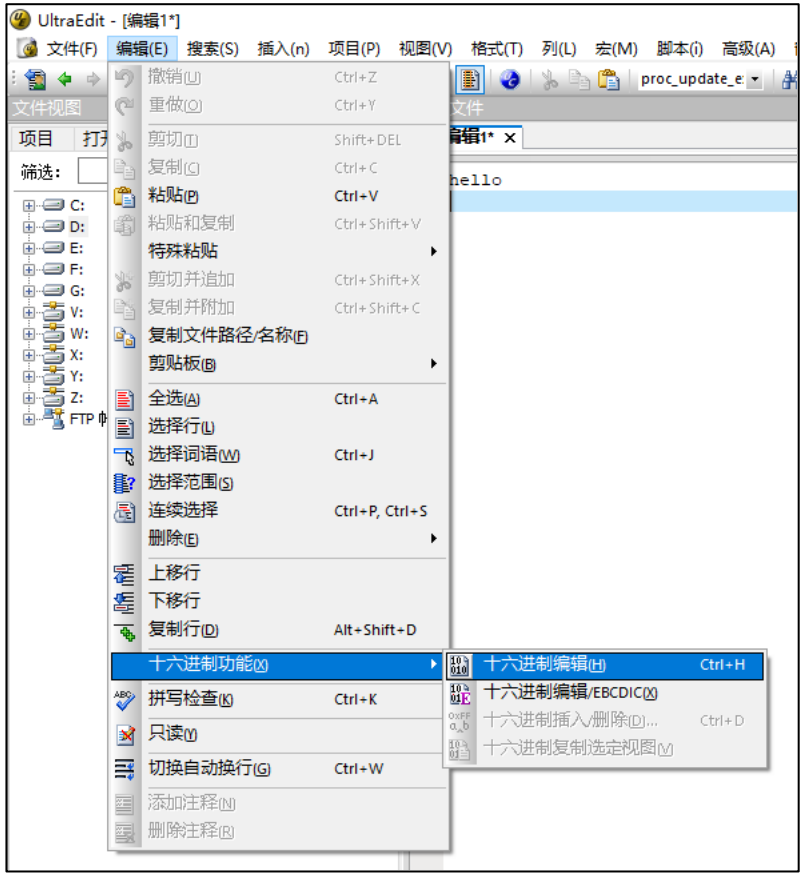
附2：附件给出的UltraEdit查看文件的16进制形式的方法（三种）



正常启动，建立文件，输入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

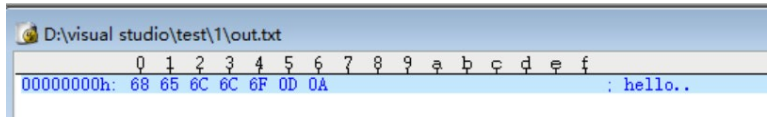
§15. 输入输出流

本页需填写答案

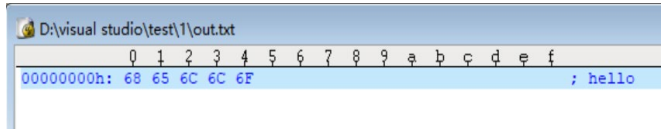
例1：十进制方式写

```
#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



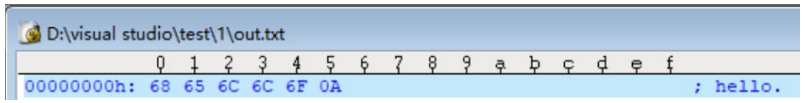
§15. 输入输出流

本页需填写答案

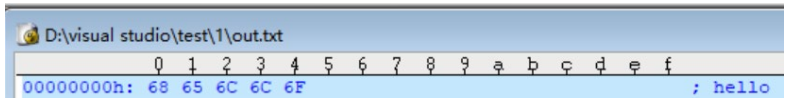
例2：二进制方式写

```
#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

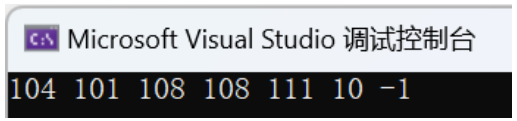
§15. 输入输出流

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___。

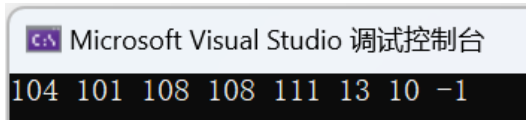
§15. 输入输出流

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和10___。

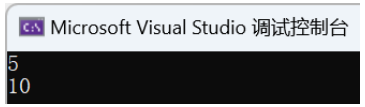
§15. 输入输出流

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;
}
```

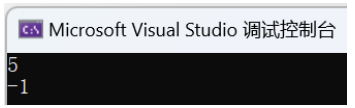
Windows下运行，输出结果是：



说明：in>>str读到__空白字符__就结束了，__换行符__还被留在缓冲区中，因此in.peek()读到了__换行符的ASCII码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：



说明：in.getline读到__换行符__就结束了，__换行符__被读掉，因此in.peek()读到了__EOF（通常是 -1）__。

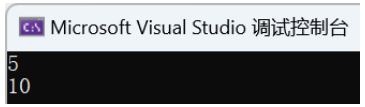
§15. 输入输出流

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;
}
```

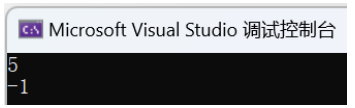
Windows下运行，输出结果是：



说明：in>>str读到__空白字符__就结束了，__换行符__还被留在缓冲区中，因此in.peek()读到了__换行符的ASCII码10__。

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

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

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

    return 0;
}
```

Windows下运行，输出结果是：



说明：in.getline读到__换行符__就结束了，__换行符__被读掉，因此in.peek()读到了__EOF（通常是 -1）__。

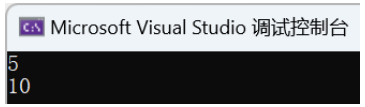
§15. 输入输出流

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;
}
```

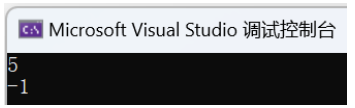
Windows下运行，输出结果是：



说明：in>>str读到__空白字符__就结束了，__换行符__还被留在缓冲区中，因此in.peek()读到了__换行符的ASCII码10__。

```
#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__换行符__就结束了，__换行符__被读掉，因此in.peek()读到了__EOF（通常是 -1）__。

§15. 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

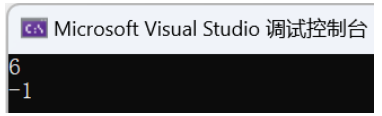
```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

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__空白字符__就结束了，__\r__还被留在缓冲区中，因此in.peek()读到了__回车的ASCII码13__。

```
#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__换行符__就结束了，_\r_被读掉，因此in.peek()读到了_EOF（通常是 -1）_。

2、strlen(str)是__6__，最后一个字符是_\r_

§15. 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？

在 C++ 中，输出流会处理字符串并在遇到 \0 时停止写入。因此，out << “ABC\0\x61\x62\x63” 只会写入ABC，而不会写入abc。由于在输出时使用了endl，这会在输出后添加了\r\n。最终是5字节：41 42 43 0D 0A

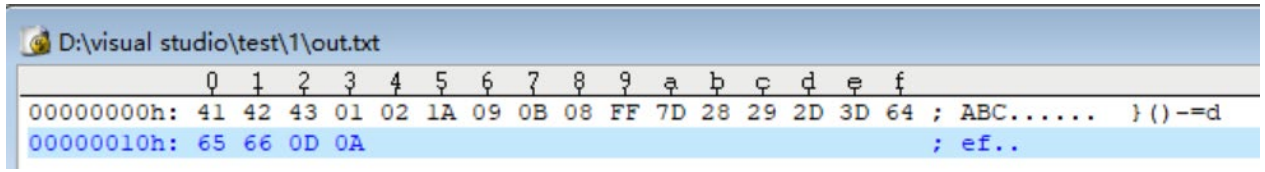
§15. 输入输出流

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：



§15. 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_____

输出的c是：_____6_____

为什么？

文件实际的字节数为20，而你计算的字符数c只有6，这是因为在读取字符的过程中遇到了文件结束符，导致读取计数停止。

```
#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：_____20_____

输出的c是：_____21_____

c的大小比文件大小大_1_，原因是：在读取到文件的最后一个字符后，in.get()会尝试读取下一个字符。虽然此时不会再有有效字符返回，但c计数器仍然会增加一次，导致最终的计数结果比实际字节数多1。

§15. 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_____

输出的c是：_____5_____

为什么？

文件大小为19字节是因为所有字符都被写入了文件，但在读取时遇到控制字符\x1A，使得读取操作提前结束，导致c的值为5。

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

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

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

    return 0;
}
```

Windows下运行，文件大小：_____19_____

输出的c是：_____5_____

为什么？

文件大小为19字节是因为所有字符都被写入了文件，但在读取时遇到控制字符\x1A，使得读取操作提前结束，导致c的值为5。

§15. 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_____
输出的c是：_____18_____

为什么？

文件大小为19字节（17个字符加2个换行符），但读取字符数为18（原始字符加换行符的处理结果），这导致输出不一致。

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

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

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

    return 0;
}
```

Windows下运行，文件大小：_____19_____
输出的c是：_____5_____

为什么？

文件大小为19字节（17个字符加2个换行符）但在读取时，由于某些字符（控制字符）可能被处理为非打印字符或被忽略，因此最终读取的字符数量被限制在5。

综合例11~例13，结论：当文件中含字符__控制字符__时，不能用十进制方式读取，而当文件中含字符__普通可见字符__时，是可以用二/十进制方式正确读取的

§15. 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_____
输出的name是：__ABCDEFGHIJKLMNOPQRSTUVWXYZ__
name[26]的值是：_____0_____
gcount()的值是：_____0_____
tellg()的值是：_____26_____
说明：in >> 方式读入字符串时，和cin方式相同，都是
读到___空格___停止，并在数组最后加入一个___\0___。

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

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

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

    return 0;
}
```

Windows下运行，文件大小：_____28_____
输出的name是：ABCDEFGHIJKLMNOPQRSTUVWXYZ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___指定的字节数___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_____。

§15. 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] = "000000000000000000000000000000";
    in.read(name, 20);
    cout << '*' << name << '*' << endl;
    cout << int(name[20]) << endl;
    cout << in.gcount() << endl;
    cout << in.tellg() << endl;
    cout << in.good() << endl;
    in.close();

    return 0;
}
```

Windows下运行，文件大小：_____26_____

输出的name是：_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] = "000000000000000000000000000000";
    in.read(name, 200);
    cout << '*' << name << '*' << endl;

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

    return 0;
}
```

Windows下运行，文件大小：_____26_____

输出的name是：_ABCDEFGHIJKLMNOPQRSTUVWXYZ000_

gcount()的值是：_____26_____

tellg()的值是：_____-1_____

good()的值是：_____0_____

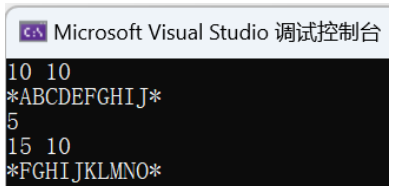
§15. 输入输出流

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

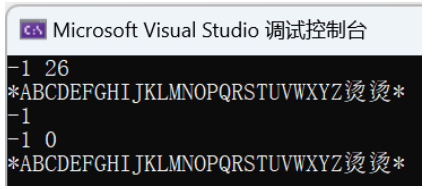
```
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //无换行符
    out.close();

    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：




```
#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：



综合左右：tellg()/gcount()/seekg()仅在___流对象处于有效状态___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。

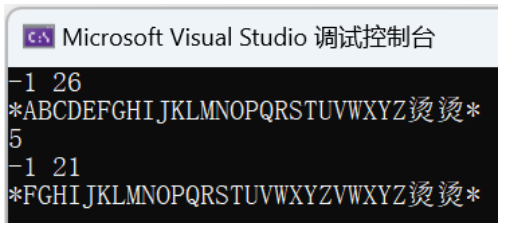
§15. 输入输出流

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：



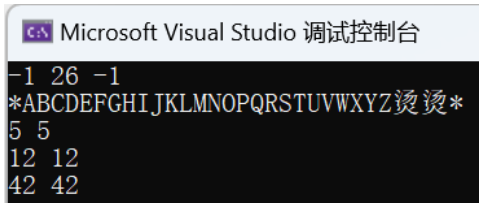
§15. 输入输出流

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：



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

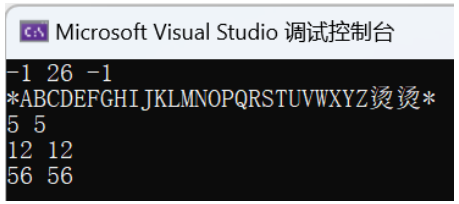
§15. 输入输出流

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：



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____总是会被移到文件末尾____

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)

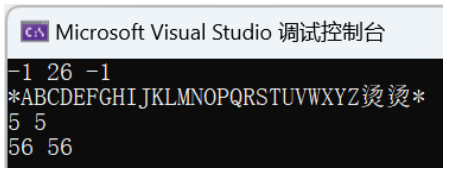
§15. 输入输出流

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：



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

§15. 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____

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_____

§15. 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____

2、加ios::ate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_____10_____

注：ate = at end

§15. 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____

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

§15. 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____

2、正常读方式打开，tellg()和peek()为__0__和___65___，

表示从文件的__结束__(开始/结束)位置读

§15. 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____
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