§13. 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

4、转换为pdf后提交

5、无特殊说明，Windows下用VS2017编译，Linux下用C++编译

6、本题在“实验报告”中提交

1850059 计1班 杨志远

§13. 输入输出流

本页需填写答案

例1：十进制方式写，在Windows/Linux下的差别

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

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

    out << "hello" << endl;

    out.close();

    return 0;
}
```

Windows下运行，out.txt是_7_字节，用UltraEdit的16进制方式打开的贴图

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

Linux下运行，out.txt是_6_字节，用UltraEdit的16进制方式打开的贴图

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

§13. 输入输出流

本页需填写答案

例2：二进制方式写，在Windows/Linux下的差别

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

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

    out << "hello" << endl;

    out.close();

    return 0;
}
```

Windows下运行，out.txt是_6_字节，用UltraEdit的16进制方式打开的贴图

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

Linux下运行，out.txt是_6_字节，用UltraEdit的16进制方式打开的贴图

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

§13. 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____。

§13. 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____。

§13. 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.get() << endl;
    in.close();

    return 0;
}
```

Windows下运行，输出结果是：

5

10

说明：in>>str读到_0x0D_就结束了，_0x0A_还被留在缓冲区中，因此in.get()读到了_0x0A_。

```
#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__0x0A__就结束了，_0x0D和0x0A_被读掉，因此in.get()读到了__eof__。

§13. 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.get() << 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，_0x0A_还被留在缓冲区中，因此in.get()读到了_0x0A_。

```
#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_0x0A_就结束了，_0x0A_被读掉，因此in.get()读到了_eof_。

§13. 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.get() << 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，_0A_还被留在缓冲区中，因此in.get()读到了_0A_。

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

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

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

    return 0;
}
```

Windows下运行，输出结果是：

5

-1

说明：in.getline读到_0A_就结束了，_0A_被读掉，因此in.get()读到了_eof_。

§13. 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.get() << 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，_0x0D_还被留在缓冲区中，因此in.get()读到了_0x0D_。

```
#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_0x0A_就结束了，_0x0A_被读掉，因此in.get()读到了_eof_。

2、strlen(str)是_6_，最后一个字符是_0x0D

§13. 输入输出流

本页需填写答案

例9：在Linux读取Windows下写的十进制文件

在Linux下运行本程序

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

int main(int argc, char *argv[])
{
    ofstream out("out.txt", ios::out);
    out << "hello\r" << endl; //模拟Windows格式
    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;
}
```

同例8右侧，未变过

```
#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;
}
```

本例说明，在Linux下读取Windows格式的文件，要注意0D的处理

6
-1
说明：
1、in.getline读到_0x0A_就结束了，_0x0A_被读掉，因此in.peek()读到了_eof_。
2、strlen(str)是_6_，最后一个字符是_0x0D_

6
-1
说明：
1、in.getline读到_0x0A_就结束了，_0x0A_被读掉，因此in.peek()读到了_eof_。
2、strlen(str)是_6_，最后一个字符是_0x0D_

§13. 输入输出流

本页需填写答案

例10：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，Linux下运行，out.txt的大小是_4_字节

为什么？

Windows下文件内容为41 42 43 0D 0A

Linux下文件内容为41 42 43 0A

Linux下没有写入0x0D，不需要0x0D表示回车

§13. 输入输出流

本页需填写答案

例11：用十进制方式写入含非图形字符(ASCII码32是空格，33-126为图形字符)，但不含\0

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

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

    return 0;
}
```

Windows下运行，out.txt的大小是_20_字节，UltraEdit的16进制显示截图为：

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

Linux下运行，out.txt的大小是_19_字节，UltraEdit的16进制显示截图为：

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

§13. 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/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_
Linux下运行，文件大小：_19_
输出的c是：_20_
为什么?Linux下没有写入0x0D，在读入0x1A后会继续读入字符。十进制方式下0x1A在Windows上有效。

```
#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_
Linux下运行，文件大小：_19__
输出的c是：_20_
c的大小比文件大小大_1_，原因是：_in读入字符时，由于in.eof()的滞后性，当读入eof时，in.eof被置为true，在下次循环条件判断时退出，因此文件末尾的eof也被计数了。二进制方式下0x1A在Windows上失效_

§13. 输入输出流

本页需填写答案

例13：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_
输出的c是：_5_
Linux下运行，文件大小：_18_
输出的c是：_18_
为什么?本程序读入eof不会计数。Linux下没有写入0D，在读入0x1A后会继续读入字符。十进制方式下1A在Windows上有效

Windows下运行，文件大小：_19_
输出的c是：_5_
Linux下运行，文件大小：_18_
输出的c是：_18_
为什么?本程序读入eof不会计数。 Linux下没有写入0D，在读入0x1A后会继续读入字符。十进制方式下0x1A在Windows上有效

§13. 输入输出流

本页需填写答案

例14：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_
Linux下运行，文件大小：_18_
输出的c是：_18_
为什么?in.get()在遇到0xFF时返回int型的255，不等于eof，因此读取方式错误

```
#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__
Linux下运行，文件大小：_18_
输出的c是：_5_
为什么？in.get()在遇到0xFF返回int型的255，被ch获取并转化为char类型，等于eof，而eof会使程序终止读取文件

综合例12~例14，结论：当文件中含字符_0x1A_时，不能用十进制方式读取，而当文件中含字符_0xFF_时，是可以用二/十进制方式正确读取的

§13. 输入输出流

本页需填写答案

例15：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_间隔符_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_
输出的name是：
ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫烫烫烫烫烫烫<q6
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_eof_停止，不在数组最后加入一个_\0_。

综合左右：gcount()仅对_非格式化输入_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_。

§13. 输入输出流

本页需填写答案

例16：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_

§13. 输入输出流

本页需填写答案

例17：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;
    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;
    return 0;
}
```

Windows下运行，输出依次是：

```
-1 26
*ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫*
-1
-1 0
*ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫*
```

综合左右：tellg()/gcount()/seekg()仅在_读取不差过文件长度的数据_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。

§13. 输入输出流

本页需填写答案

例18：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();

    return 0;
}
```

Windows下运行，输出依次是：

```
-1 26
*ABCDEFGHIJKLMNOPQRSTUVWXYZ烫烫*
5
-1 21
*FGHIJKLMNOPQRSTUVWXYZVWXYZ烫烫*
```