§13.输入输出流

要求：
1、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读写差异
2、需完成的页面，右上角有标注，直接在本文件上作答，用蓝色写出答案即可
3、转换为pdf后提交
4、无特殊说明，Windows下用VS2017编译，Linux下用C++编译

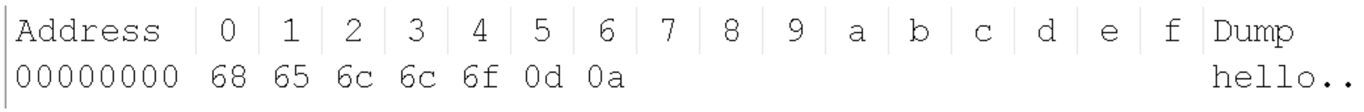
§13.输入输出流

本页需填写答案

例1：十进制方式写，在Windows/Linux下的差别

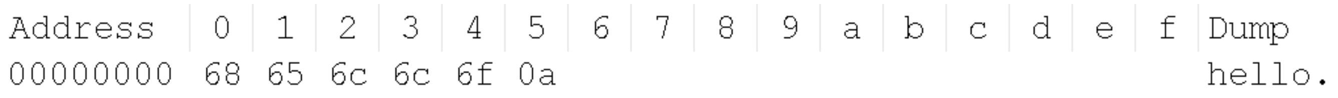
```
#include<iostream>
#include<fstream>
using namespace std;

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



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000  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



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000  68 65 6c 6c 6f 0a                               hello.
```

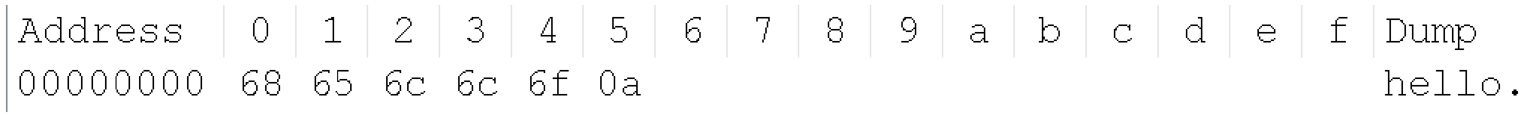
§13. 输入输出流

本页需填写答案

例2：二进制方式写，在Windows/Linux下的差别

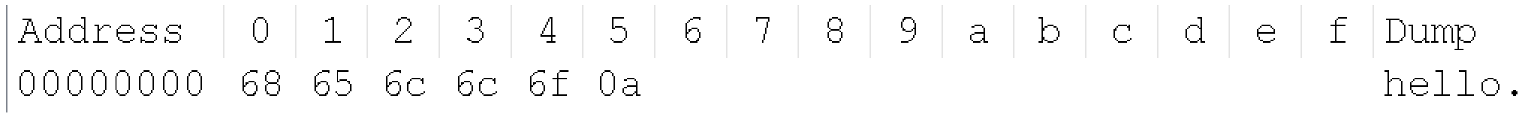
```
#include<iostream>
#include<fstream>
using namespace std;

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



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000  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



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000  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;
    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;
    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的十进制方式下被当做__2__个字符处理，值是__13,10____。

§ 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;

    return 0;
}
```

Windows下运行，输出结果是：

```
5
10
```

说明：in>>str读到_回车_就结束了，_换行_还被留在缓冲区中，因此in.get()读到了__换行\n(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;

    return 0;
}
```

Windows下运行，输出结果是：

```
5
-1
```

说明：in.getline读到__回车_就结束了，__回车__被读掉，因此in.peek()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;

    return 0;
}
```

Windows下运行，输出结果是：

```
5
10
```

说明：in>>str读到_回车_就结束了，_换行_还被留在缓冲区中，因此in.get()读到了__换行\n(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;

    return 0;
}
```

Windows下运行，输出结果是：

```
5
-1
```

说明：in.getline读到__回车_就结束了，__回车__被读掉，因此in.peek()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;

    return 0;
}
```

Windows下运行，输出结果是：

```
5
10
```

说明：in>>str读到_回车_就结束了，_换行_还被留在缓冲区中，因此in.get()读到了__换行\n(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;

    return 0;
}
```

Windows下运行，输出结果是：

```
5
-1
```

说明：in.getline读到__回车_就结束了，__回车__被读掉，因此in.peek()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;

    return 0;
}
```

Windows下运行，输出结果是：

```
5
13
```

说明：in>>str读到_回车_就结束了，_回车换行_还被留在缓冲区中，因此in.get()读到了_回车\r(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;

    return 0;
}
```

Windows下运行，输出结果是：

```
6  -1
```

说明：in.getline读到__换行_就结束了，__换行__被读掉，因此in.peek()读到了__EOF__。
2、strlen(str)是_6_，最后一个字符是_\r_

§13.输入输出流

本页需填写答案

例9：在Linux读取Wwindows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;

    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;

    return 0;
}
```

本例说明，在Linux下读取Windows格式的文件，要注意0D的处理

Linux下运行，输出结果是：

6 -1

说明：

1、in.getline读到__回车__就结束了，_回车__被读掉，因此in.peek()读到了_EOF_。

2、strlen(str)是_6，最后一个字符是_\r__

Windows下运行，输出结果是：

6 -1

说明：

1、in.getline读到__回车__就结束了，_回车__被读掉，因此in.peek()读到了_EOF_。

2、strlen(str)是_6，最后一个字符是_\r__

§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？
因为Windows下endl代表 回车\r(13) 与 换行\n(10)
然而Linux 下endl代表 换行\n(10)

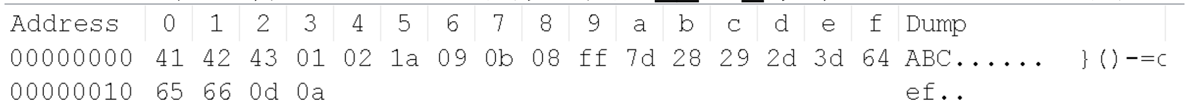
§13.输入输出流

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

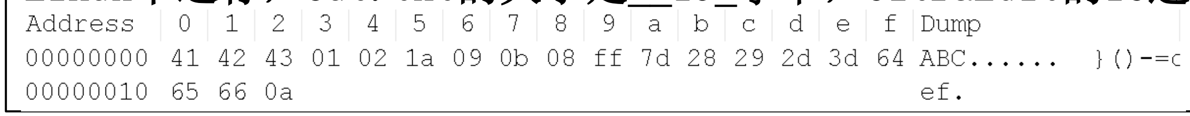
```
#include <iostream>
#include <fstream>
using namespace std;

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：



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 41 42 43 01 02 1a 09 0b 08 ff 7d 28 29 2d 3d 64 ABC......  }()-=d
00000010 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：



```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 41 42 43 01 02 1a 09 0b 08 ff 7d 28 29 2d 3d 64 ABC......  }()-=d
00000010 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;

    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下不能将CRTL+Z作为eof()的结束标志

```
#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;

    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：将最后的-1读入以后，再一次循环后，eof()才能跳出

§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;

    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？与前一张ppt不同，CTRL+Z可以认为是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\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;

    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？与本张ppt左侧示例类似，只是用一个char类型的变量ch接受get()得到的结果

§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)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;

    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\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;

    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？在内存中的-1(0xFF)并不能被识别成与CTRL+Z相同的效果，即不能作为eof()的结束符

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？将文件中的-1(0xFF)读取给，某个变量，通过变量判断是否为-1可被eof()识别

综合例12~例14，结论：当文件中含字符CTRL+Z(\x1A)时，不能用十进制方式读取，而当文件中含字符\XFF时，是可以用二/十进制方式正确读取的（emmmm...，世上本无事，你偏要找点事出来）