§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

<朱世轩 计2 1752528>

§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

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

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

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

```
            0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
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;
    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__0A____。

§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的二进制方式下被当做__2__个字符处理，值是__0D 0A____。

§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_0D_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);
    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_0D_就结束了_0D 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;
}
```

```
#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

10

说明：in>>str读到_0D___就结束了，_0A__还被留在缓冲区中，因此in.get()读到了__0A___。

Windows下运行，输出结果是：

5

-1

说明：in.getline读到_0D_就结束了，0D 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__0D__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;

    return 0;
}
```

Windows下运行，输出结果是：

5

-1

说明：in.getline读到_0D_就结束了，0D 0A被读掉，因此in.get()读到了__0A___。

§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__0D__就结束了，_0D_还被留在缓冲区中，因此in.get()读到_0D__。

```
#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：1、in.getline读到__0D__就结束了，__0D_0A_被读掉，因此in.peek()读到_EOF_。

2、strlen(str)是_6_，最后一个字符是_\0_

§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__0D__就结束了，0D 0A被读掉，因此in.peek()读到了_EOF_。

2、strlen(str)是_6_，最后一个字符是_\0_

Windows下运行，输出结果是：

6

-1

说明：1、in.getline读到_0D_就结束了0D 0A被读掉，因此in.get()读到了_EOF_。

2、strlen(str)是_6_，最后一个字符是_\0_

§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？

ABC为3字节，读到\0结束，windows中endl为\r\n占2字节，而linux为\n占1字节。所以windows下为5字节，linux下为4字节

§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：

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC......  }()
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：

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 41 42 43 01 02 1A 09 0B 08 FF 7D 28 29 2D 3D 64 ; ABC......  }()
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;

    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?windows下读到\x1A直接结束，而linux下文件结束标志不为ctrl+z，正常读取

```
#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 | 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：遇到EOF使c多累加了一次，而EOF不计入文件大小

§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？windows下读到\x1A直接结束，而linux下文件结束标志不为ctrl+z，正常读取

```
#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;
    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？windows下读到\x1A直接结束，而linux下文件结束标志不为ctrl+z，正常读取。判断了读到EOF结束，所以c与文件大小相同

§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字节_____
输出的c是：____18_____
Linux下运行，文件大小：_____18字节_____
输出的c是：_____18_____
为什么?正确读取时\xFF作为正常字符处理，可以正常读取

Windows下运行，文件大小：____19字节_____
输出的c是：___5_____
Linux下运行，文件大小：_____18字节_____
输出的c是：____5_____
为什么?将\xFF赋值给字符后作为EOF处理，读到EOF时c中止累加

综合例12~例14，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）