



§ . 基础知识题 – 文件的基本操作(C方式)

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读写差异，掌握与文件有关的函数的正确用法
- 3、需完成的页面，右下角有标注，直接在本文件上作答，**用蓝色写出答案**即可
★ 运行结果**允许**截图后贴在文档中，内容不要相互重叠即可
- 4、无特殊说明，Windows下用VS2019编译(**后缀为.c**)，如果要换成其他编译器，可能需要自行修改头文件适配
★ 除题目明确指定编译器外，缺省使用VS2019即可
★ 部分代码编译时**有warning**，不影响概念理解，**可以忽略**
- 5、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
- 6、转换为pdf后在“实验报告”中提交 (**12.27前**)

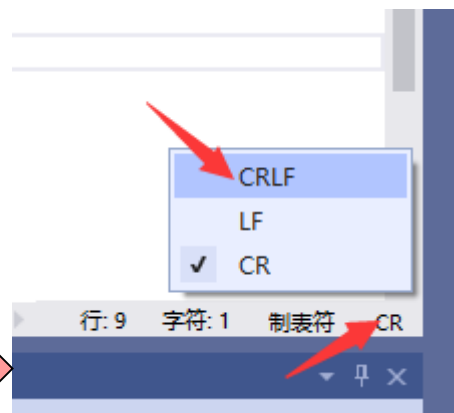
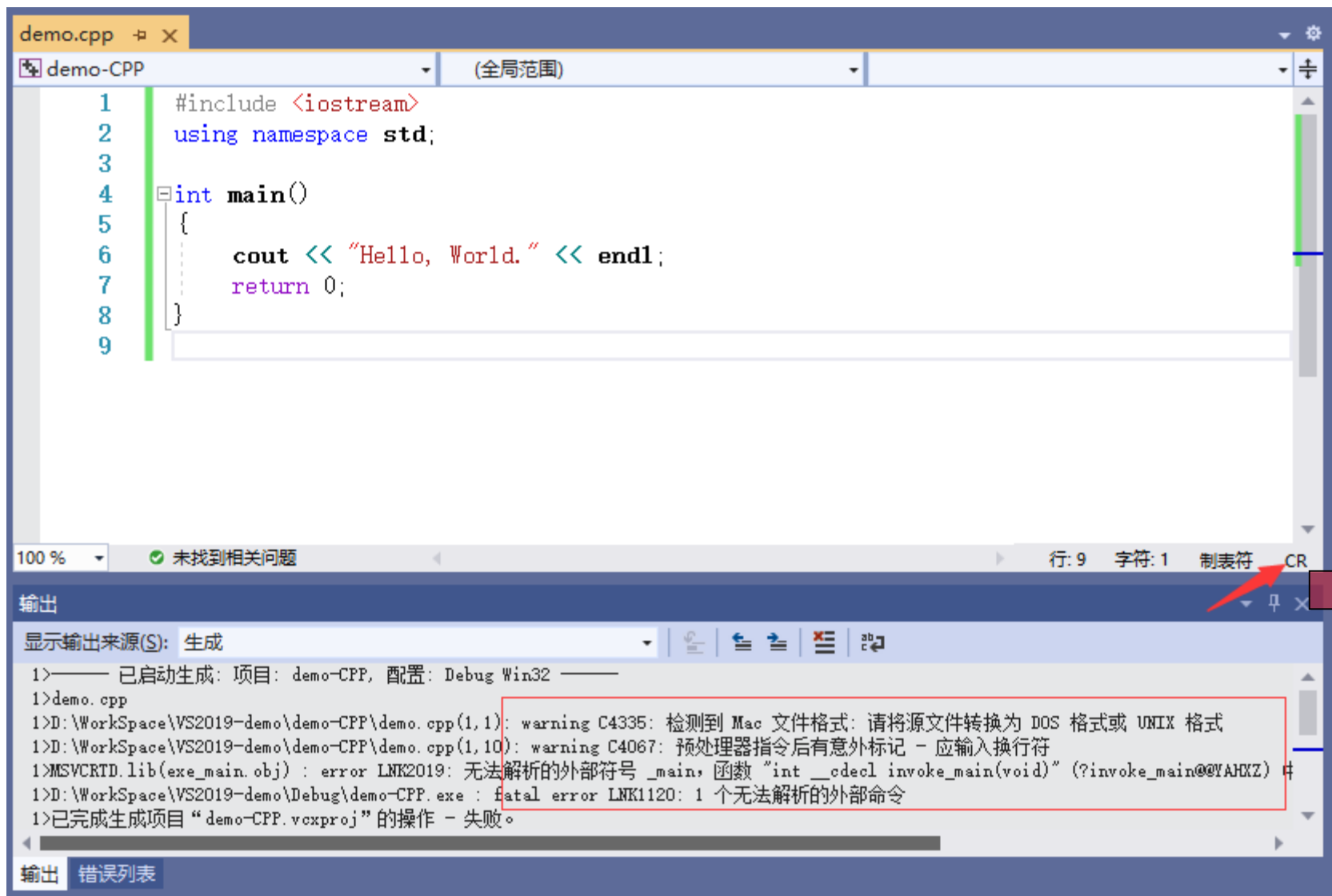


§. 基础知识题 - 文件的基本操作(C方式)

注意:

附1: 用WPS等其他第三方软件打开PPT, 将代码复制到VS2019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

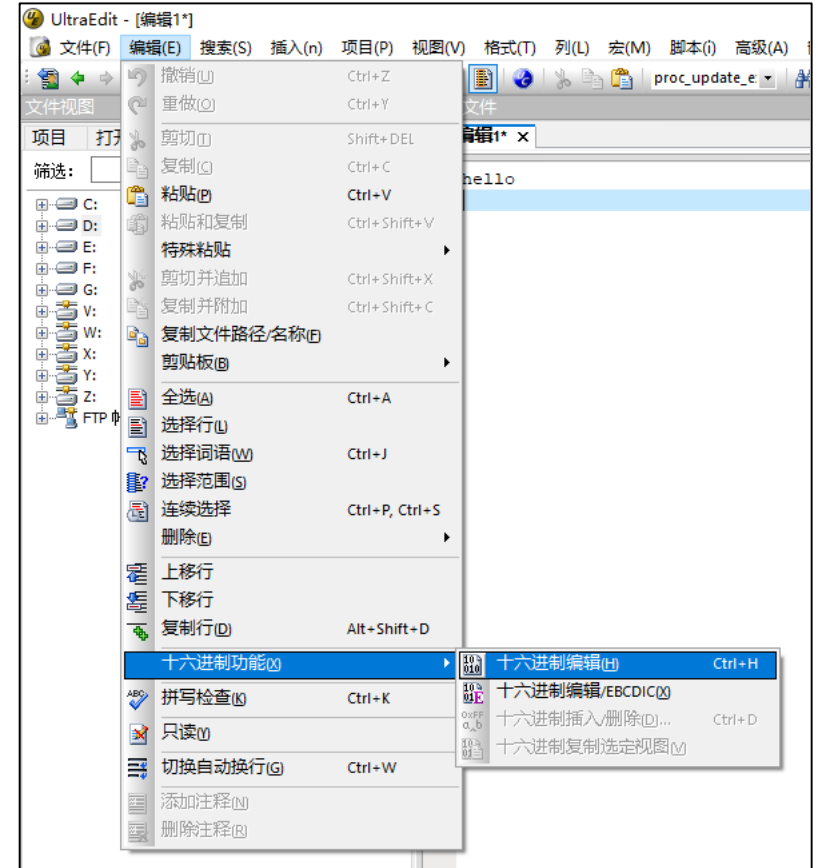
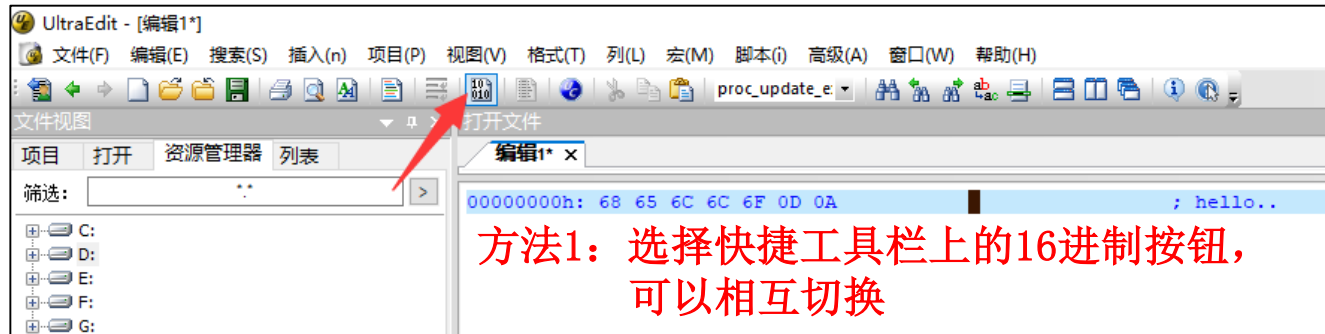
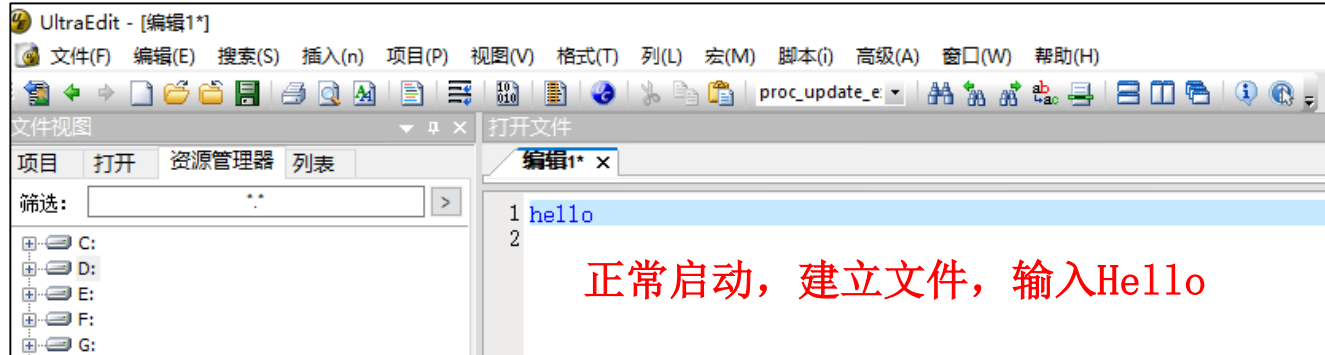




§. 基础知识题 – 文件的基本操作(C方式)

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



§. 基础知识题 – 文件的基本操作(C方式)

例1: 十进制方式写

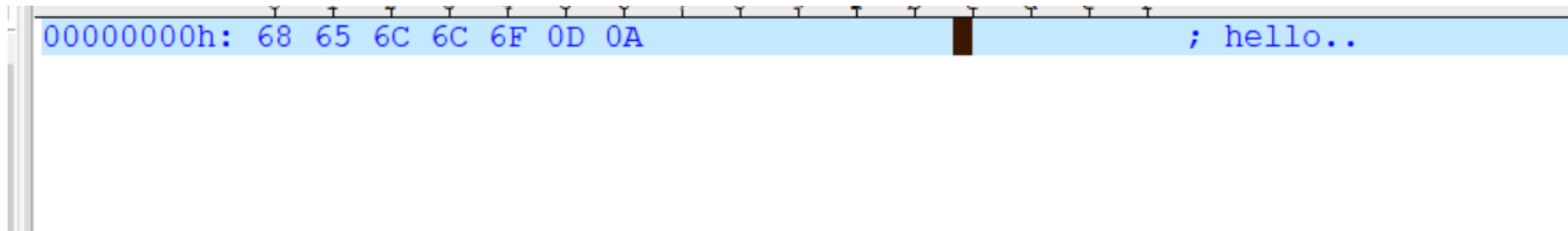
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    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



本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例2: 二进制方式写

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "wb");
    fprintf(fout, "hello\n");
    fclose(fout);

    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.

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例3: 十进制方式写, 十进制方式读, 0D0A在Windows下的表现

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    fin = fopen("out.txt", "r");
    while (!feof(fin))
        printf("%d ", fgetc(fin));
    printf("\n");
    fclose(fin);
    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__。

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例4: 十进制方式写, 二进制方式读, 0D0A在Windows下的表现

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout, *fin;

    fout = fopen("out.txt", "w");
    fprintf(fout, "hello\n");
    fclose(fout);

    fin = fopen("out.txt", "rb");
    while (!feof(fin))
        printf("%d ", fgetc(fin));
    printf("\n");
    fclose(fin);
    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 10_。

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

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>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "r"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "r"); fgets(str, sizeof(str), fin); //课上未讲，自行查阅 printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); 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>6 -1</div>
说明：fscanf() 读到 <u>o</u> 就结束了， <u>\n</u> 还被留在缓冲区中，因此fgetc() 读到了 <u>\n</u> 。	说明：fgets() 读到 <u>\n</u> 就结束了， <u>\n</u> 被读掉，因此fgetc() 读到了 <u>EOF</u> 。



§. 基础知识题 – 文件的基本操作(C方式)

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>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "wb"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "r"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "wb"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "r"); fgets(str, sizeof(str), fin); //课上未讲，自行查阅 printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); 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>6 -1</div>
说明：fscanf() 读到 <u>o</u> 就结束了， <u>\n</u> 还被留在缓冲区中，因此fgetc() 读到了 <u>\n</u> 。	说明：fgets() 读到 <u>\n</u> 就结束了， <u>\n</u> 被读掉，因此fgetc() 读到了 <u>EOF</u> 。



§. 基础知识题 – 文件的基本操作(C方式)

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

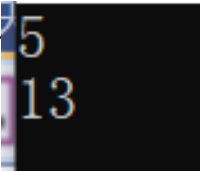
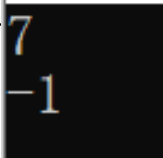
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "wb"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "rb"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "wb"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "rb"); fgets(str, sizeof(str), fin); //课上未讲，自行查阅 printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); 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>6 -1</div>
说明：fscanf() 读到__o__就结束了，__\n__还被留在缓冲区中，因此fgetc() 读到了__\n__。	说明：fgets() 读到__\n__就结束了，__\n__被读掉，因此fgetc() 读到了__EOF__。

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

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>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "rb"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "rb"); fgets(str, sizeof(str), fin); //课上未讲，自行查阅 printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); fclose(fin); return 0; }</pre>
<p>Windows下运行，输出结果是：</p> <p>说明：fscanf() 读到 <u>o</u> 就结束了，<u>\r</u> 还被留在缓冲区中，因此fgetc() 读到了 <u>\r</u>。</p>	<p>Windows下运行，输出结果是：</p> <p>说明：fgets() 读到 <u>\n</u> 就结束了，<u>\n</u> 被读掉，因此fgetc() 读到了 <u>EOF</u>。</p>

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例9: 用十进制方式写入含\0的文件, 观察文件长度

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\0\x61\x62\x63");
    fclose(fout);

    return 0;
}
```

Windows下运行, out.txt的大小是__3__字节, 为什么?
fprintf中遇到\0就停止了, 字符串结束的标志, 所以只有ABC写入到文件中。

用UltraEdit的16进制方式显示的截图:

00000000h: 41 42 43 ; ABC

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

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

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fout;

    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\x1\x2\x1A\t\v\b\xff\175()-=def");
    fclose(fout);

    return 0;
}
```

Windows下运行(VS有warning) , out.txt的大小是__18__字节, 为什么?

分字符处理

用UltraEdit的16进制方式显示的截图:

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

VS的waring是:

: warning C4819: 该文件包含不能在当前代码页(0)中表示的字符。请将该文件保存为 Unicode 格式以防止数据丢失

VS的哪个字符导致了warning?

`\xff`

本页需填写答案



§ . 基础知识题 – 文件的基本操作(C方式)

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>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\x1A\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "r"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\x1A\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "rb"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>
<p>Windows下运行，文件大小： <u>17字节</u> 输出的c是： <u>6</u></p> <p>为什么？ CTRL+Z是文件结束的标志，文件结束不进入while循环了，读了6个字符。</p>	<p>Windows下运行，文件大小： <u>17字节</u> 输出的c是： <u>18</u> c的大小比文件大小大<u>1字节</u>，原因是：读完f之后，接下来的fin指向文件结束符，但并未读取，所以文件并未结束，进入while循环，fgetc结束符eof-1之后文件结束，所以多读了一个。</p>

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例11: (改) 只允许修改while循环, 使“c=xx”的输出 (左侧: \x1A前字符数, 右侧: 与文件大小一致)

<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\x1A\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "r"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\x1A\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "rb"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>

另: 思考一下, 上周cin的成员函数作业, CTRL+Z的ASCII是多少?

-1



§ . 基础知识题 – 文件的基本操作(C方式)

例12: 用十进制方式写入含\xFF(十进制255/-1, EOF的定义是-1)的文件, 并用十/二进制读取

<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\xFF\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "r"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\xFF\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "rb"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>
Windows下运行, 文件大小: <u>17字节</u> 输出的c是: <u>18</u>	Windows下运行, 文件大小: <u>17字节</u> 输出的c是: <u>18</u>
综合例11~例12, 结论: 当文件中含字符_0x1A_ (0x1A/0xFF) 时, 不能用十进制方式读取, 而当文件中含字符_0xFF_ (0x1A/0xFF) 时, 是可以用二/十进制方式正确读取的	

本页需填写答案



§ . 基础知识题 – 文件的基本操作(C方式)

例12: (改) 只允许修改while循环, 使“c=xx”的输出与文件大小一致

<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\xFF\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "r"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABC\x1\x2\xFF\t\v\b\175()--def"); fclose(fout); fin = fopen("out.txt", "rb"); int c=0; while(!feof(fin)) { fgetc(fin); c++; } printf("c=%d\n", c); fclose(fin); return 0; }</pre>
Windows下运行, 文件大小: <u>17字节</u> 输出的c是: <u>17</u>	Windows下运行, 文件大小: <u>17字节</u> 输出的c是: <u>17</u>

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例13: 比较格式化读和read()读的区别, 并观察ftell在不同读入方式时值的差别

<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABCDEFGHJKLMNOPQRSTUVWXYZ\n"); fclose(fout); fin = fopen("out.txt", "rb"); char name[30]; fscanf(fin, "%s", name); printf("%s*\n", name); printf("%d\n", name[26]); printf("%d\n", ftell(fin)); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABCDEFGHJKLMNOPQRSTUVWXYZ\n"); fclose(fout); fin = fopen("out.txt", "rb"); char name[30]; fread(name, sizeof(name), 1, fin); printf("%s*\n", name); printf("%d\n", name[26]); printf("%d\n", ftell(fin)); fclose(fin); return 0; }</pre>
<p>Windows下运行, 文件大小: <u>28字节</u> 输出的name是 <u>ABCDEFGHJKLMNOPQRSTUVWXYZ</u> name[26]的值是: <u>0</u> ftell的值是: <u>26</u> 说明: fscanf方式读入字符串时, 和scanf方式相同, 都是读到 <u>' \n' </u> 停止, 并在数组最后加入一个 <u>' \0' </u>。</p>	<p>Windows下运行, 文件大小: <u>28字节</u> 输出的name是: <u>ABCDEFGHJKLMNOPQRSTUVWXYZ加乱码</u> name[26]的值是: <u>13</u> ftell的值是: <u>28</u> 说明: fread() 读入时, 是读到 <u>' \n' </u> 停止, 不在数组最后加入一个 <u>' \0' </u>。</p>

不要截图, 手填, 确定乱码的地方可以填“乱码”

本页需填写答案



§ . 基础知识题 – 文件的基本操作(C方式)

例14: 比较read()读超/不超过文件长度时的区别, 并观察ftell()/feof()的返回值

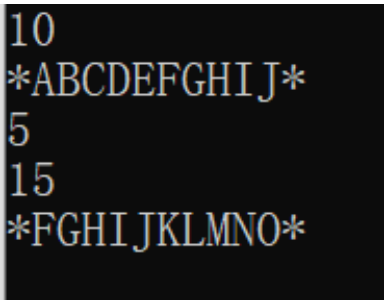
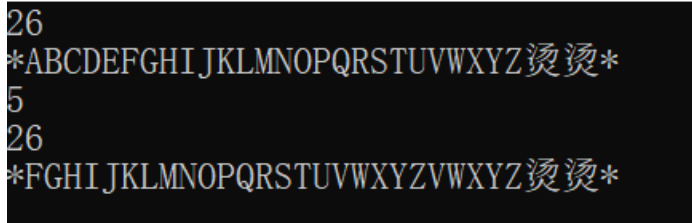
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABCDEFGHJKLMNOPQRSTUVWXYZ");//无\n fclose(fout); fin = fopen("out.txt", "rb"); char name[30] = "00000000000000000000000000000000"; fread(name, 20, 1, fin); printf("%s*\n", name); printf("%d\n", name[20]); printf("%d\n", ftell(fin)); printf("%d\n", feof(fin)); fclose(fin); return 0; }</pre>	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABCDEFGHJKLMNOPQRSTUVWXYZ");//无\n fclose(fout); fin = fopen("out.txt", "rb"); char name[30] = "00000000000000000000000000000000"; fread(name, 200, 1, fin); printf("%s*\n", name); printf("%d\n", ftell(fin)); printf("%d\n", feof(fin)); fclose(fin); return 0; }</pre>
Windows下运行, 文件大小: <u>26字节</u> 输出的name是: <u>ABCDEFGHJKLMNOPRST0000000000</u> name[20]的值是: <u>48</u> ftell的值是: <u>20</u> feof的值是: <u>0</u>	Windows下运行, 文件大小: <u>26字节</u> 输出的name是: <u>ABCDEFGHJKLMNOPQRSTUVWXYZ000</u> ftell的值是: <u>26</u> feof的值是: <u>1</u>

本页需填写答案



§ . 基础知识题 – 文件的基本操作(C方式)

例15: 使用fseek()移动文件指针, 观察ftell()不同情况下的返回值

<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABCDEFGHJKLMNOPQRSTUVWXYZ"); //无换行符 fclose(fout); fin = fopen("out.txt", "rb"); char name[80]; fread(name, 10, 1, fin); printf("%d\n", ftell(fin)); name[10] = '\0'; printf("%s\n", name); fseek(fin, -5, SEEK_CUR); printf("%d\n", ftell(fin)); fread(name, 10, 1, fin); printf("%d\n", ftell(fin)); name[10] = '\0'; printf("%s\n", name); fclose(fin); return 0; }</pre> 	<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "ABCDEFGHJKLMNOPQRSTUVWXYZ"); //无换行符 fclose(fout); fin = fopen("out.txt", "rb"); char name[80]; fread(name, 30, 1, fin); printf("%d\n", ftell(fin)); name[30] = '\0'; printf("%s\n", name); fseek(fin, 5, SEEK_SET); printf("%d\n", ftell(fin)); fread(name, 30, 1, fin); printf("%d\n", ftell(fin)); name[30] = '\0'; printf("%s\n", name); fclose(fin); return 0; }</pre> 
<p>Windows下运行, 输出依次是: (可截图, 需对结果做分析) fread1次读10个字符, 正好是ABCDEFGHJIJ, 然后此时文件指针距离文件开始位置10字节, 所以ftell的值为10, name中是ABCDEFGHJIJ所以输出也是这个, 调用fseek函数让文件指针向前移动5个字节, 所以距离文件开始5个字节, 从此时开始读10个字母FGHIJKLMNOP, 之后ftell是15, 最后输出刚才的字母</p>	<p>Windows下运行, 输出依次是: (可截图, 需对结果做分析) fread读30个大于26, 所以此时距离文件开头26, 输出26, 之后26到30还有4个没赋值的位置, 是烫乱码, 所以输出两个烫, 一个汉字两个字节, 之后文件指针从开始向后移动5个, 移到指向F, 之后读30个, 但是只能读到Z, 所以从A开始用从F到Z重新覆盖, 之后就形成了name数组。</p>

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例16: fread的返回值理解

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;
    char buf[80];

    fp = fopen("in.txt", "r");
    int ret = fread(buf, 26, 1, fp);
    printf("ret=%d\n", ret);
    fclose(fp);

    return 0;
}
```

准备工作: 在当前目录下建in.txt文件,
写入A..Z共26个字母, 不要加回车
确定文件大小为26字节!!!

fread的第2/3参数:

原26, 1, ret=1
换1, 26, ret=26
换13, 2, ret=2
换2, 13, ret=13
换80, 1, ret=0
换1, 80, ret=26
换15, 2, ret=1
换2, 15, ret=13

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例17: fread用于二进制/十进制方式打开的文件时的返回值理解

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;
    char buf[80];

    fp = fopen("in.txt", "r");
    int ret = fread(buf, 1, 80, fp);
    printf("ret=%d\n", ret);
    fclose(fp);

    return 0;
}
```

准备工作：当前目录下建in.txt文件，写多行

例：abc
123
xyz

注：1、考虑到字符集问题，不要中文
2、文件总大小不超过50字节
3、最后一行加不加回车均可

文件编辑完成后，Windows右键菜单查看文件属性，能看到大小是__13__字节。

运行左侧程序，打印的ret=__11__

将“r”改为“rb”，再次运行，打印的ret=__13__

两次运行结果不一样的原因是：‘\r’的计算



§. 基础知识题 – 文件的基本操作(C方式)

例18: fwrite返回值理解

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;
    char buf[80]="abcdefghijklmnopqrstuvwxyz";

    fp = fopen("out.txt", "w");
    int ret = fwrite(buf, 26, 1, fp);
    printf("ret=%d\n", ret);
    fclose(fp);

    return 0;
}
```

fwrite的第2/3参数:

原26, 1, ret=__1__, 文件大小__26_
换1, 26, ret=__26_, 文件大小__26_
换13, 2, ret=__2_, 文件大小__26_
换2, 13, ret=__13_, 文件大小__26_
换80, 1, ret=__1_, 文件大小__80_
换1, 80, ret=__80_, 文件大小__80_
换15, 2, ret=__2_, 文件大小__30_
换2, 15, ret=__15_, 文件大小__30_

本页需填写答案



§. 基础知识题 – 文件的基本操作(C方式)

例19: fwrite用于二进制/十进制方式打开的文件时的返回值理解

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *fp;
    char buf[80]="abc\n123\nxyz\n";

    fp = fopen("out.txt", "w");
    int ret = fwrite(buf, 1, strlen(buf), fp);
    printf("ret=%d\n", ret);
    fclose(fp);

    return 0;
}
```

运行左侧程序，打印的ret=__12__，
Windows右键菜单查看文件属性，大小是__15__字节。

将“w”改为“wb”，再次运行，打印的
ret=__12__，Windows右键菜单查看
文件属性，大小是__12__字节。

两次运行打印的ret一样，但文件属性中看到的文件大小不一样的原因是：
__还是\n的问题，w是文本模式写，
那么遇到\n则会在文件中出现的是
\r\n两个，所以第一个多了3个\r多了3字节，而下面的二进制写，不会多出\r，则为12字节__。