



§ 15. 输入输出流进阶 – C语言的文件操作

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差异，掌握与文件有关的函数的正确用法
- 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
 - ★ 部分代码编译时有warning，不影响概念理解，可以忽略
- 3、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
- 4、**11月3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
- ★ 无特殊说明，Windows下用VS2022编译(**后缀为.c**)

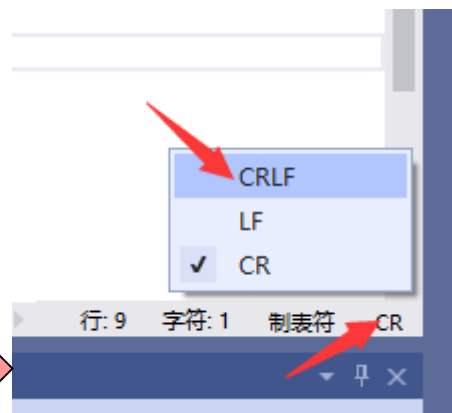
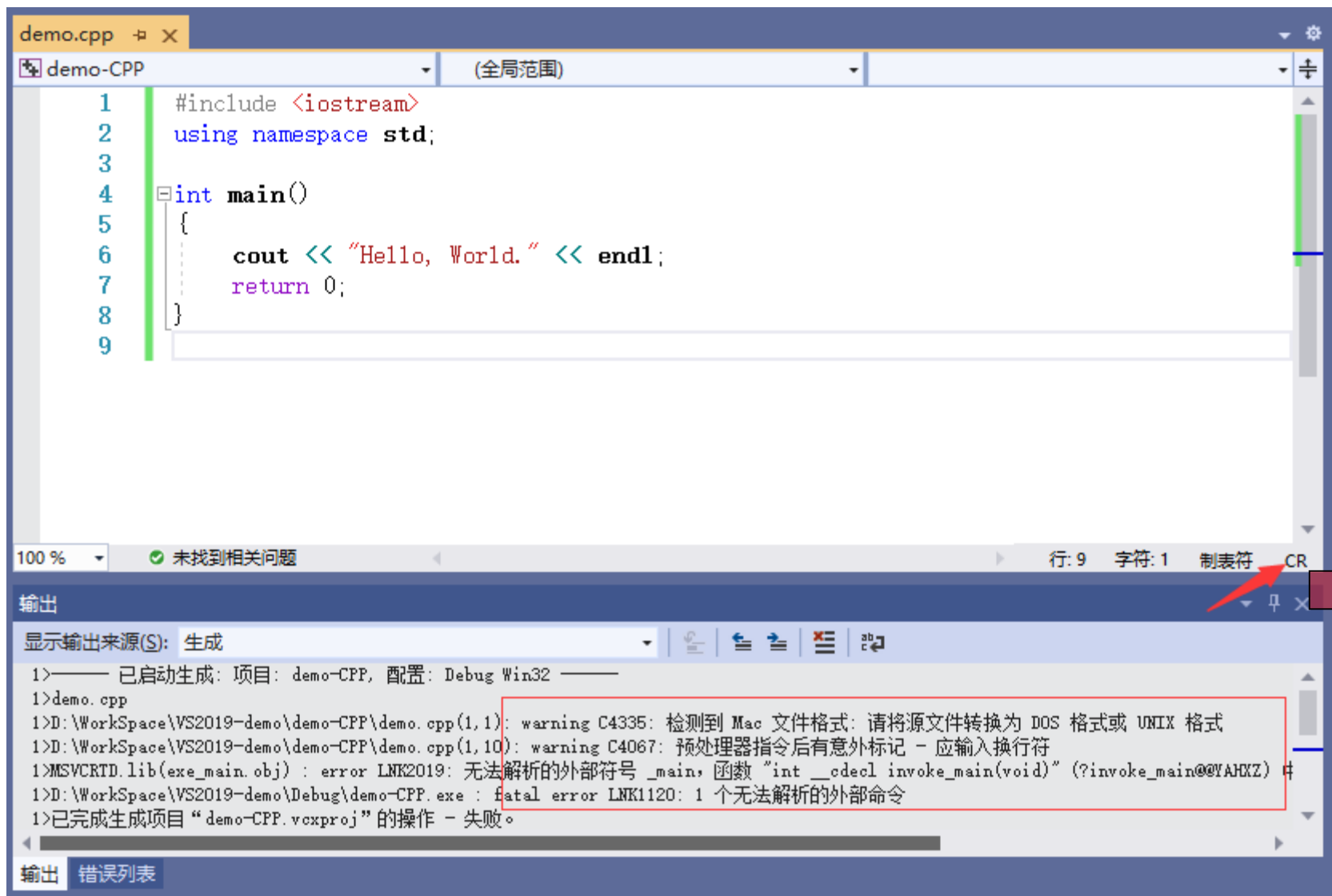


§ 15. 输入输出流进阶 – C语言的文件操作

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

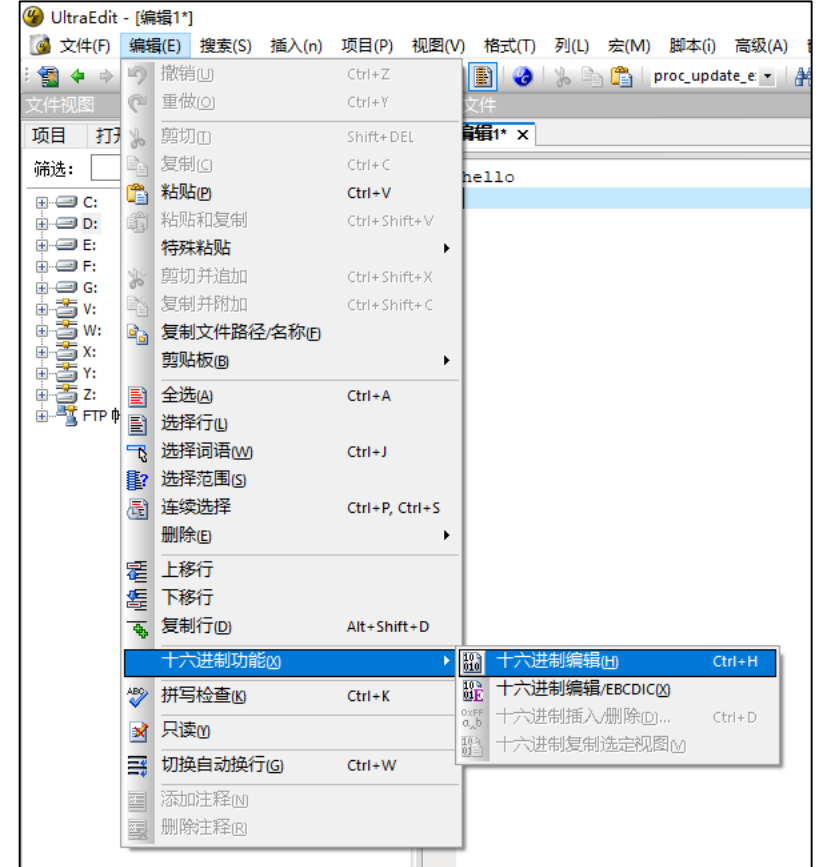
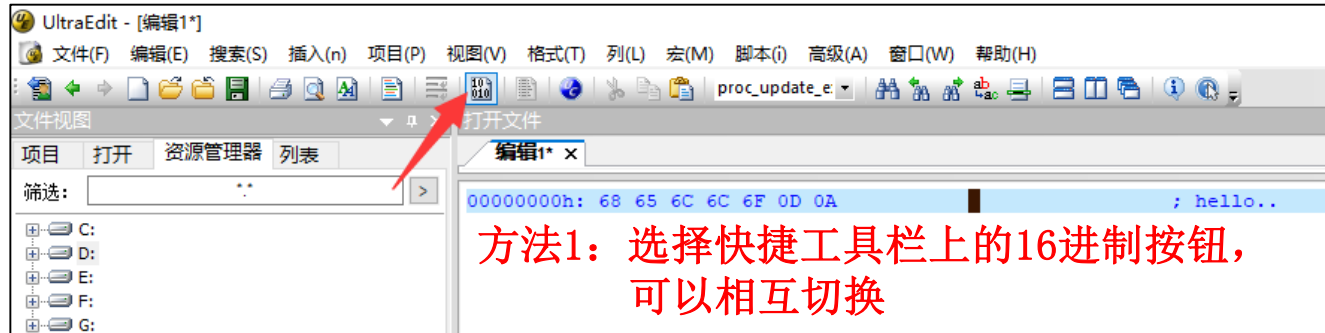
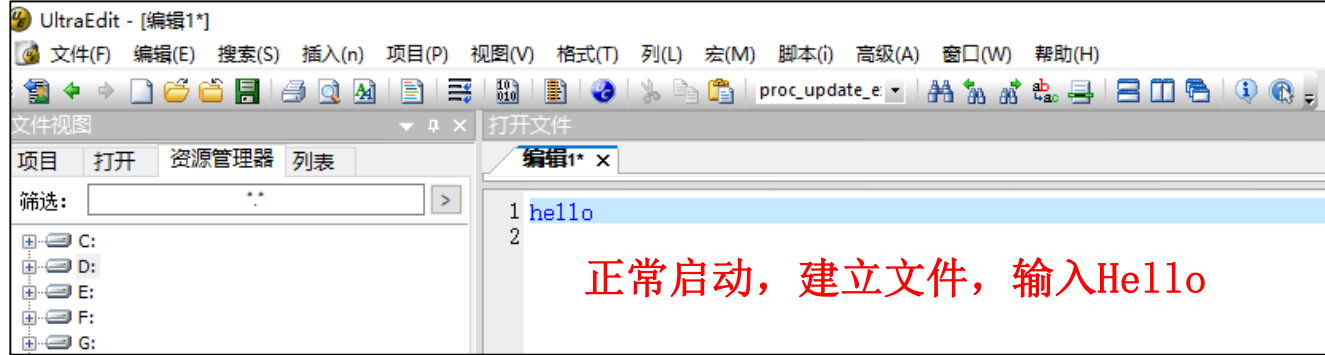




§ 15. 输入输出流进阶 – C语言的文件操作

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



§ 15. 输入输出流进阶 – C语言的文件操作

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

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

本页需填写答案



§ 15. 输入输出流进阶 - C语言的文件操作

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



§ 15. 输入输出流进阶 – C语言的文件操作

例3: 十进制方式写, 十进制方式读, 0D0A在Windows下的表现

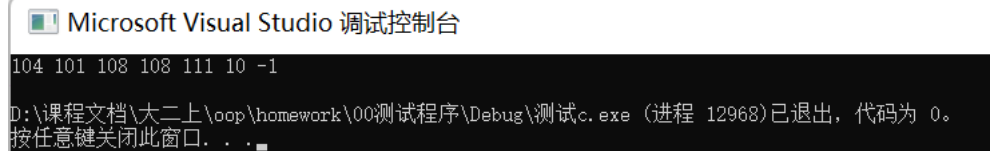
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

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:



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



§ 15. 输入输出流进阶 – C语言的文件操作

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
D:\课程文档\大二上\oop\homework\00测试程序\Debug\测试c.exe (进程 1340)已退出，代码为 0。
按任意键关闭此窗口...
```



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



§ 15. 输入输出流进阶 – C语言的文件操作

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： 	Windows下运行，输出结果是： 
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_ 0A ____。	说明：fgets() 读到_ ’ \n’ ____就结束了， _ ’ \n’ ____被读掉，因此fgetc() 读到了_ - 1 (EOF) ____。

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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>
<p>Windows下运行，输出结果是：</p> <div><pre>5 10</pre></div> <p>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 _0A_。</p>	<p>Windows下运行，输出结果是：</p> <div><pre>6 -1</pre></div> <p>说明：fgets()读到‘ \n’ 就结束了， _ ‘ \n’ _被读掉，因此fgetc()读到了____ _ -1 (EOF) _。</p>

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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


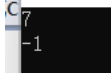
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "wb"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "rb"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); 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>
<p>Windows下运行，输出结果是：</p> 	<p>Windows下运行，输出结果是：</p> 
<p>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__0A__。</p>	<p>说明：fgets() 读到__\n__就结束了，__\n__ 被读掉，因此fgetc() 读到了_-1 (EOF)___。</p>

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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>Windows下运行，输出结果是：</p> 
<p>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 0D。</p>	<p>说明：fgets() 读到 __\\n__ 就结束了， __\\n_ 被读掉，因此fgetc() 读到了 __-1 (EOF) 。</p>

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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>

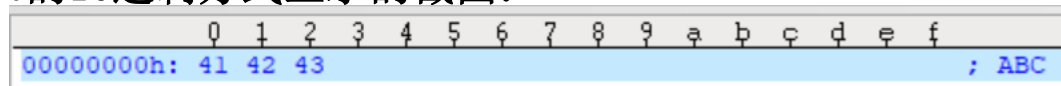
int main()
{
    FILE *fout;

    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？
输出空字符，即输出到\0时输出终止。

用UltraEdit的16进制方式显示的截图：



本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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?

\x后的部分字符将会成为转义字符, 合并为一个字符, 计算后得到18字节

用UltraEdit的16进制方式显示的截图:

	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..... }()-=d
00000010h:	65	66															; ef

VS的warning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



§ 15. 输入输出流进阶 – C语言的文件操作

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>
Windows下运行，文件大小：_ 17字节_____ 输出的c是：_6_____	Windows下运行，文件大小：__17字节_____ 输出的c是：__18_____
为什么?读到0x1A时，识别为EOF，读取结束	c的大小比文件大小大，原因是：_当feof遇到EOF时，返回的0，只有读到下一个字符时才返回1，所以多读了一个字节_____

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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>	<div>改:</div> <pre>while(!feof(fin)) { int end = fgetc(fin); if(c == -1) break; }</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>	<div>改:</div> <pre>while(!feof(fin)) { int end = fgetc(fin); if(end == -1)break; c++; }</pre>
Windows下运行，文件大小：__17字节__ 输出的c是：____5____	Windows下运行，文件大小：__17字节__ 输出的c是：____17____		

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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: __17字节__ 输出的c是: __18__	Windows下运行, 文件大小: __17字节__ 输出的c是: __18__
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



§ 15. 输入输出流进阶 – C语言的文件操作

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>	
<div>Windows下运行, 文件大小: <u> 17字节 </u> 输出的c是: <u> 17 </u></div>		<div>Windows下运行, 文件大小: <u> 17字节 </u> 输出的c是: <u> 17 </u></div>	

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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>
Windows下运行, 文件大小: __28字节__ 输出的name是: _ABCDEFGHJKLMNOPQRSTUVWXYZ_ name[26]的值是: __0__ ftell的值是: __26__ 说明: fscanf方式读入字符串时, 和scanf方式相同, 都是读到_\n__停止, 并在数组最后加入一个_\0__。	Windows下运行, 文件大小: __28字节__ 输出的name是: __ABCDEFGHJKLMNOPQRSTUVWXYZ乱码__ name[26]的值是: __13__ ftell的值是: __28__ 说明: fread()读入时, 是读到_\n__停止, 不在数组最后加入一个_\0__。

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



§ 15. 输入输出流进阶 – C语言的文件操作

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: 28字节 输出的name是: _ABCDEFGHJKLMNOPQRST0000000000_ name[20]的值是: 48 ftell的值是: 20 feof的值是: 0	Windows下运行, 文件大小: 26字节 输出的name是: _ABCDEFGHJKLMNOPQRSTUVWXYZ0000_ ftell的值是: 26 feof的值是: 1

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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)</p> <div><pre>10 *ABCDEFGHIJ* 5 15 *FGHIJKLMNO*</pre></div> <p>Fread读取大小为十字节的一块, 此时指针在10字节处, ftell返回10, name得到前10个字符, fseek前移5字节, ftell返回5, fread读取之后10字节的内容到name中, fseek前移10字节, ftell返回15, 在输出name内容</p>	<p>Windows下运行, 输出依次是: (可截图, 需对结果做分析)</p> <div><pre>26 *ABCDEFGHJKLMNOPQRSTUVWXYZ烫烫* 5 26 *FGHIJKLMNOPQRSTUVWXYZVWXYZ烫烫*</pre></div> <p>Fread读取大小为30字节的一块, 指针指向文件末尾即26字节处, 因为在30字节处加\0, 26后面部分未知, 为乱码, fseek从开始后移5字节, 在读取30字节, 同理文件指针指向最后, fread读取之后30字节的内容到name中, ftell返回15, 在输出name内容</p>

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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



§ 15. 输入输出流进阶 – C语言的文件操作

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____28____字节。

运行左侧程序，打印的ret=____23____

将“r”改为“rb”，再次运行，打印的ret=____28____

两次运行结果不一样的原因是：____二进制



§ 15. 输入输出流进阶 – C语言的文件操作

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
字节____
换1, 26, ret=__26__, 文件大小_26
字节____
换13, 2, ret=_2__, 文件大小_26字
节____
换2, 13, ret=_13__, 文件大小_26
字节____
换80, 1, ret=_1__, 文件大小_80字
节____
换1, 80, ret=_80__, 文件大小_80
字节____
换15, 2, ret=_2__, 文件大小_30字
节____
换2, 15, ret=__15__, 文件大小_30
字节____

本页需填写答案



§ 15. 输入输出流进阶 – C语言的文件操作

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：
_____。