



§ 15. 输入输出流 - C语言的文件操作

要求:

- 1、安装UltraEdit软件（**附件已给，版本虽旧，但够用**），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
 - ★ 已安装VSCode的也可通过相关插件进行16进制方式的查看（**VSCode某种情况下会自动做格式转换或字符集转换，要注意!!!**）
 - ★ 也可以使用其它编辑软件，但**不建议**NotePad++
- 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（**后缀必须为.c**）
 - ★ 如果要换成其他编译器，可能需要自行修改头文件适配（**不强制要求Linux，但建议试一试**）
 - ★ 部分代码编译时有**warning**，不影响概念理解，**可以忽略**
- 4、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
- 5、转换为pdf后提交
- 6、**11月14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**

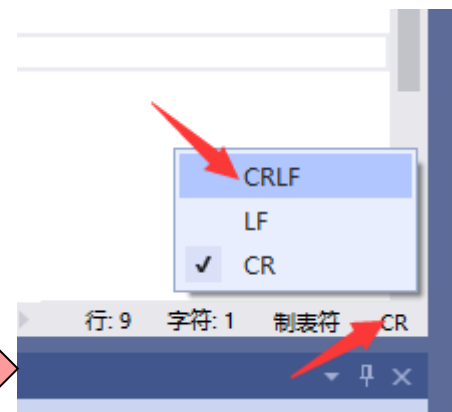
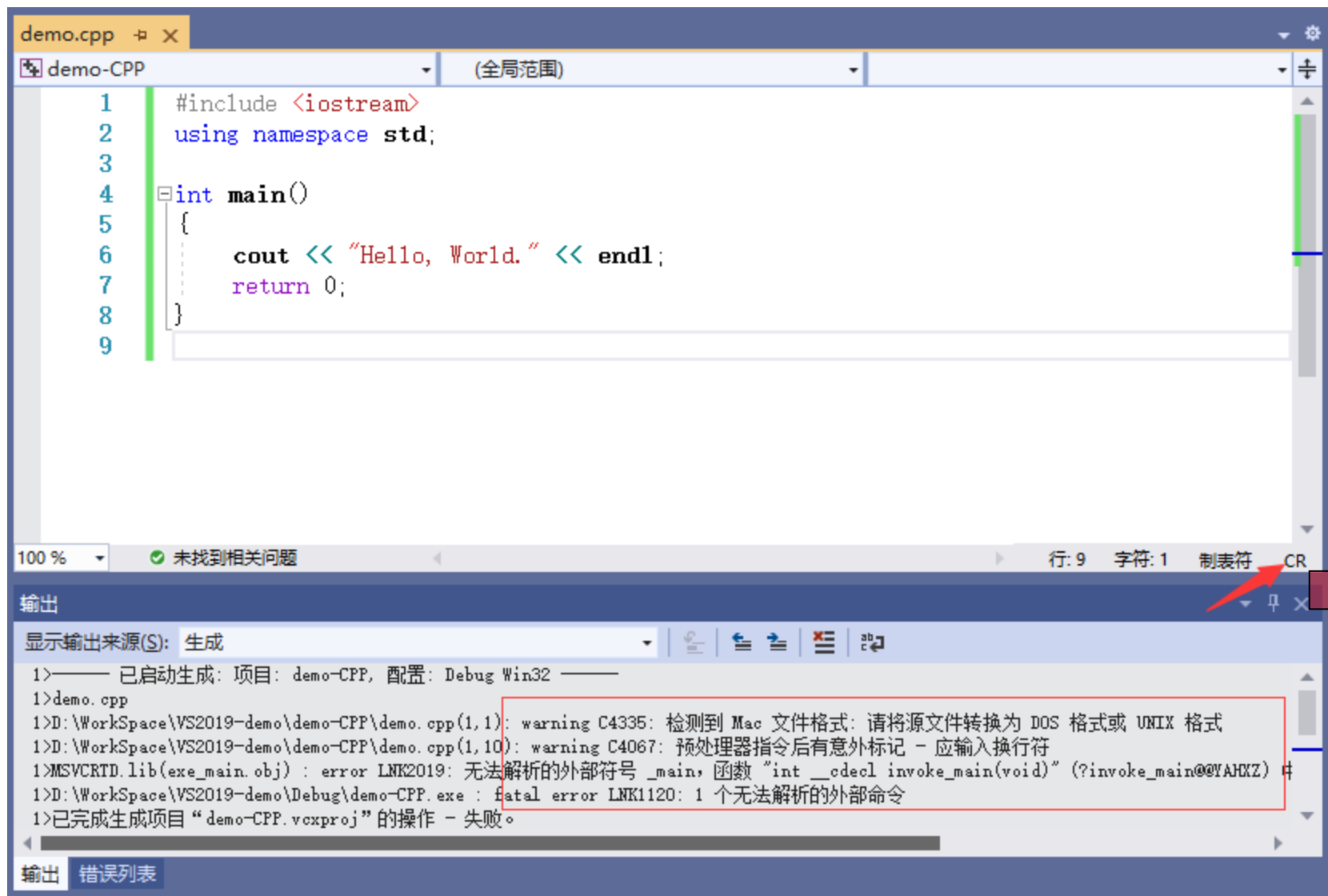


§ 15. 输入输出流 - C语言的文件操作

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

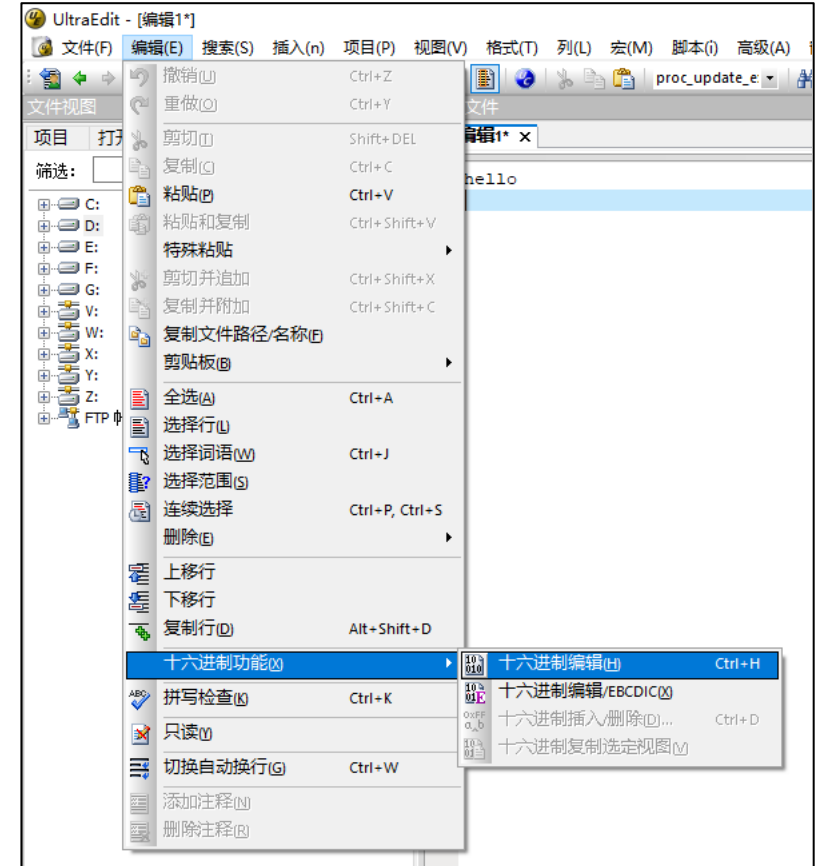
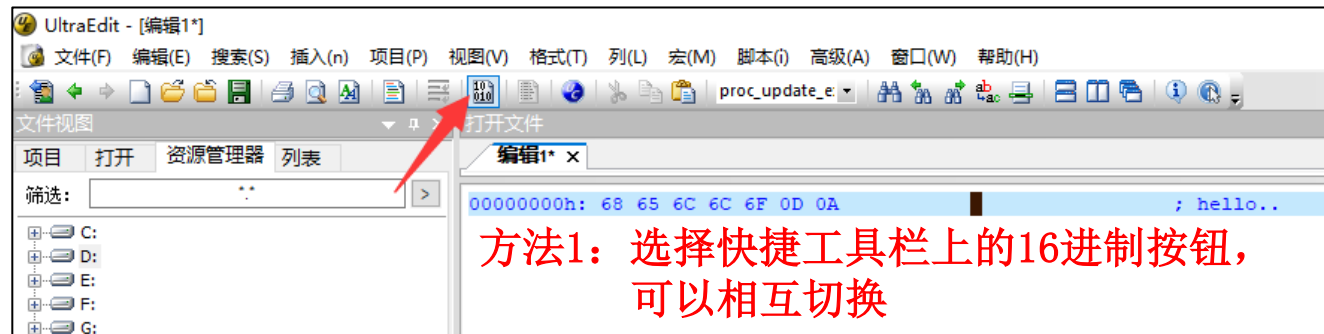
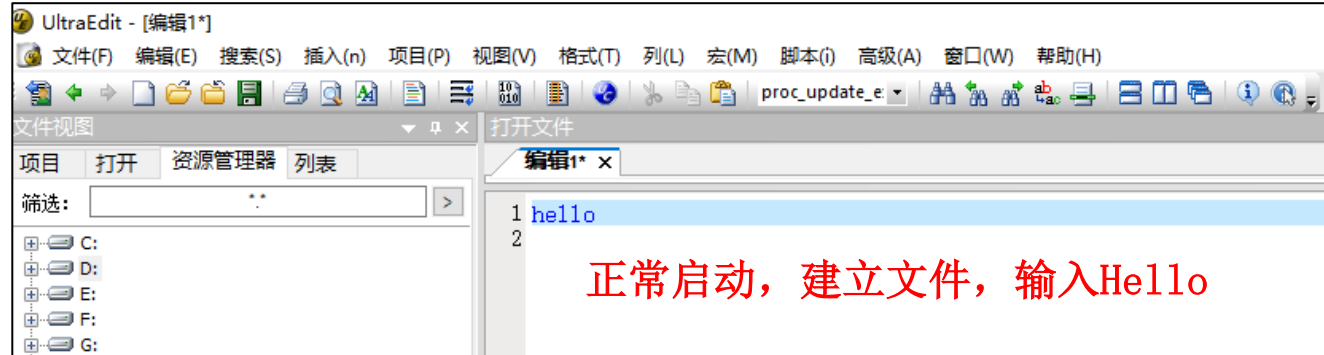




§ 15. 输入输出流 - C语言的文件操作

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 - C语言的文件操作

例1: 十进制方式写

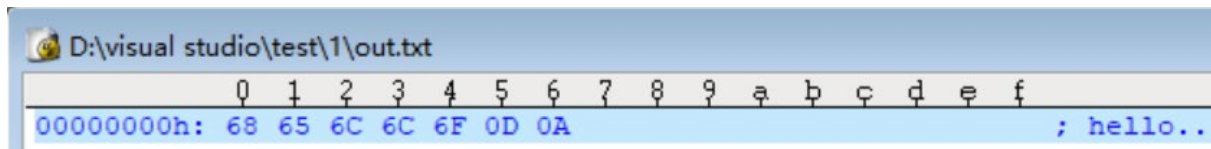
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

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



本页需填写答案



§ 15. 输入输出流 - C语言的文件操作

例2: 二进制方式写

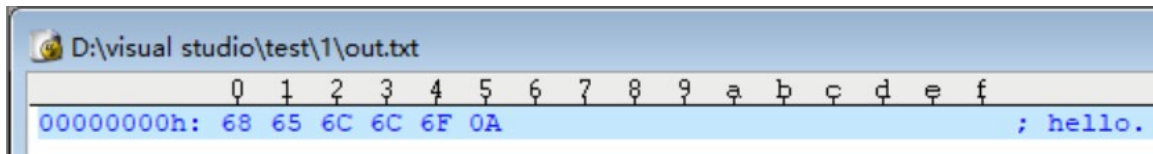
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

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



本页需填写答案



§ 15. 输入输出流 - 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

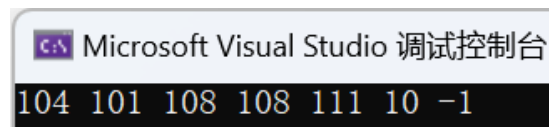
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

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:



Microsoft Visual Studio 调试控制台

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



§ 15. 输入输出流 - 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:

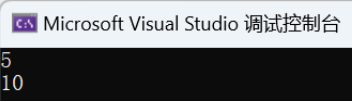
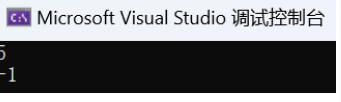
说明: 0D 0A在Windows的二进制方式下被当做__2__个字符处理, 值是__13和10__。

本页需填写答案



§ 15. 输入输出流 - 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

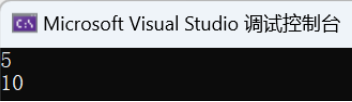
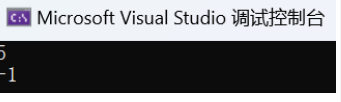
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "w"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "r"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); 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>
<p>Windows下运行，输出结果是：</p>  <p>说明：fscanf() 读到__空白字符__就结束了，__换行符__还被留在缓冲区中，因此fgetc() 读到了__换行符的ASCII码10__。</p>	<p>Windows下运行，输出结果是：</p>  <p>说明：fgets() 读到__换行符__就结束了，__换行符__被读掉，因此fgetc() 读到了__EOF (通常是 -1)__。</p>

本页需填写答案



§ 15. 输入输出流 - 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

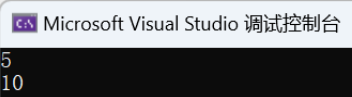
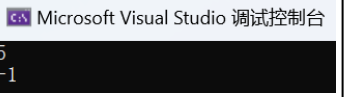
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "wb"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "r"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); 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>  <p>说明：fscanf() 读到__空白字符__就结束了，__换行符__还被留在缓冲区中，因此fgetc() 读到了__换行符的ASCII码10__。</p>	<p>Windows下运行，输出结果是：</p>  <p>说明：fgets() 读到__换行符__就结束了，__换行符__被读掉，因此fgetc() 读到了__EOF (通常是 -1)__。</p>

本页需填写答案



§ 15. 输入输出流 - 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

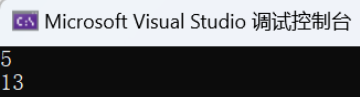
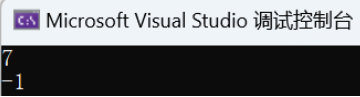
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> #include <string.h> int main() { FILE *fout, *fin; fout = fopen("out.txt", "wb"); fprintf(fout, "hello\n"); fclose(fout); char str[80]; fin = fopen("out.txt", "rb"); fscanf(fin, "%s", str); printf("%d\n", strlen(str)); printf("%d\n", fgetc(fin)); 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>说明：fscanf() 读到__空白字符__就结束了，__换行符__还被留在缓冲区中，因此fgetc() 读到了__换行符的ASCII码10__。</p>	<p>Windows下运行，输出结果是：</p>  <p>说明：fgets() 读到__换行符__就结束了，__换行符__被读掉，因此fgetc() 读到了__EOF (通常是 -1)__。</p>

本页需填写答案



§ 15. 输入输出流 - 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>说明：in>>str读到__空白字符__就结束了，__\r__还被留在缓冲区中，因此in.peek()读到了__回车的ASCII码13__。</p>	<p>Windows下运行，输出结果是：</p>  <p>说明：fgets()读到__换行符__就结束了，__换行符__被读掉，因此fgetc()读到了__EOF (通常是 -1)__。</p>

本页需填写答案



§ 15. 输入输出流 - 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;

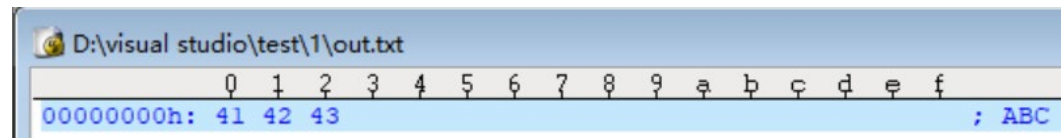
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\0\x61\x62\x63");
    fclose(fout);

    return 0;
}
```

Windows下运行, out.txt的大小是__3__字节, 为什么?

在 C 中, 输出流会处理字符串并在遇到 \0 时停止写入。因此, fprintf(fout, "ABC\0\x61\x62\x63") 只会写入ABC, 而不会写入abc。

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 - C语言的文件操作

例10: 用十进制方式使用了非 ASCII 字符 \xFF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;

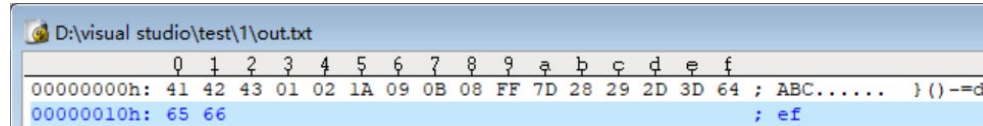
    fout = fopen("out.txt", "w");
    fprintf(fout, "ABC\x1\x2\x1A\t\v\b\xff\175()-=def");
    fclose(fout);

    return 0;
}
```

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

在 fprintf 中写入了字符串 “ABC\x1\x2\x1A\t\v\b\xff\175()-=def” 长度为18

用UltraEdit的16进制方式显示的截图:



VS的warning是:

warning C4819: 该文件包含不能在当前代码页(936)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?

非ASCII字符\xff

本页需填写答案



§ 15. 输入输出流 - 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>
<p>Windows下运行，文件大小： <u>17</u> 输出的c是： <u>6</u></p> <p>为什么？ 文件实际的字节数为17，而计算的字符数c只有6，这是因为在读取字符的过程中遇到了文件结束符，导致读取计数停止。</p>	<p>Windows下运行，文件大小： <u>17</u> 输出的c是： <u>18</u></p> <p>c的大小比文件大小大<u>1</u>，原因是：在读取到文件的最后一个字符后，feof()会尝试读取下一个字符。虽然此时不会再有有效字符返回，但c计数器仍然会增加一次，导致最终的计数结果比实际字节数多1。</p>

本页需填写答案



§ 15. 输入输出流 - 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: <pre>int c=0; int ch; while ((ch = fgetc(fin)) != EOF) { if (ch == '\x1A') break; c++; }</pre></div>
Windows下运行, 文件大小: <u>17</u> 输出的c是: <u>6</u>	<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: <pre>int c=0; int ch; while ((ch = fgetc(fin)) != EOF) { c++; }</pre></div>
Windows下运行, 文件大小: <u>17</u> 输出的c是: <u>18</u>	

本页需填写答案



§ 15. 输入输出流 - 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: <u>17</u> 输出的c是: <u>18</u>	Windows下运行, 文件大小: <u>17</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



§ 15. 输入输出流 - 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>
Windows下运行, 文件大小: <u>17</u> 输出的c是: <u>18</u>	Windows下运行, 文件大小: <u>17</u> 输出的c是: <u>18</u>

本页需填写答案



§ 15. 输入输出流 - 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: <u>28</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>空格</u> 停止, 并在数组最后加入一个 <u>\0</u> 。	Windows下运行, 文件大小: <u>28</u> 输出的name是: <u>ABCDEFGHJKLMNOPQRSTUVWXYZ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>指定的字节数</u> 停止, 不在数组最后加入一个 <u>\0</u> 。

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 - 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: <u>28</u> 输出的name是: <u>ABCDEFGHJKLMNOPQRST000000000</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>28</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



§ 15. 输入输出流 - 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> <p>该代码首先写入字符串 A-Z 到文件中, 再通过二进制模式读取。第一次读取10个字符 ("ABCDEFGHIJ") 后, 文件指针位置为10。然后, 指针向回移动5个位置到5, 再次读取10个字符得到"FGHIJKLMNO", 此时指针位置变为15。</p>	<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>在程序中, fread 读取了文件中的 26 个字符, 但由于 name 数组大小为 80, 剩余的数组部分没有被初始化。当打印 name 数组时, 未读取部分的未初始化数据被当作字符串输出, 因此出现了如 *ABCDEFGHJKLMNOPQRSTU VWXYZ烫烫* 的结果。</p>
<p>Windows下运行, 输出依次是: (可截图, 需对结果做分析)</p>	<p>Windows下运行, 输出依次是: (可截图, 需对结果做分析)</p>

本页需填写答案



§ 15. 输入输出流 - 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 - 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__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：在十进制模式下，Windows会处理换行符并将其转换为\r\n，导致读取的字节数少于文件的实际字节数。在二进制模式下，没有对换行符做任何转换。



§ 15. 输入输出流 - 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__
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__



§ 15. 输入输出流 - 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：__十进
制模式会将换行符 \n 转换为 \r\n，而
二进制模式不会进行此转换__。